Analysis of the Dynamics of Cognitive Processes

Bosse, T.

2005

document version
Publisher's PDF, also known as Version of record

Analysis of the Dynamics of Cognitive Processes

SIKS Dissertation Series No. 2005-15.

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

VRIJE UNIVERSITEIT

Analysis of the Dynamics of Cognitive Processes

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. T. Sminia,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op woensdag 23 november 2005 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Tibor Bosse

geboren te Abcoude

promotoren: prof.dr. J. Treur
prof.dr. C.M. Jonker

Acknowledgements

I started my PhD project in January 2002. From that moment to the completion of my thesis, I have known all kinds of experiences that make research challenging, varying from moments of enormous frustration to moments of extraordinary joy. Therefore, an appropriate metaphor to describe my PhD project would be a long road, sometimes going uphill and sometimes going downhill, but eventually leading to a new destination. Obviously, the parts of the road going downhill were rather easy to pass through, but the parts going uphill were much harder. Especially in such moments, when it is hard to make any progress, it is helpful when there are people around that can support, in the broadest sense of the word. Fortunately, in my case such people have always been present when I needed them, in the form of colleagues, friends or family members. Therefore, I hereby wish to let all these people know how extremely grateful I am for their support. Thus, in terms of the methodology used throughout this thesis, the following property holds: "To all persons who ever helped me with my PhD project, I will be grateful forever". Or, in a more formal notation, the property acknowledgement(Tibors_life) holds, where:

acknowledgement(γ:Γ) ≡
∀p:PERSON ∀t:T
  [state(γ, t) |= helpful_for_with(p, Tibor, PhD_project)] ⇒
    ∀t'>t state(γ, t') |= grateful_to(Tibor, p)]

I will not name each person for whom this property holds explicitly, for two reasons. Firstly, because it would take way too much space, and secondly, because it brings along the risk of forgetting people. Nevertheless, there are a few persons who deserve special attention. For that reason, I will address them below.

In the first place I wish to thank my promotors, Jan Treur and Catholijn Jonker, for everything they did for me during the past four years. Before I started my project, several friends warned me that promotors are usually very busy people, who hardly have any time for the actual supervision of their students. Although the first assertion is indeed true, the second turned out to be completely wrong. Whenever I got stuck with my research, my promotors immediately made time for me to answer my questions. As a result, together we have had an endless amount of inspiring discussions about various aspects of my research (and beyond), in which I could profit maximally from their knowledge. More specifically, I would like to thank Jan for bringing me in contact with the area of cognitive science, and for teaching me how to deal with the painful aspects of academic life (such as rejections of papers). In addition, many special thanks to Catholijn for improving my skills in writing papers and giving presentations, by forcing me to always place the highest demands on my own work (even when writing this acknowledgement!).

Next, I would like to thank all my colleagues (or should I say friends?) from the AI department, and from the Agent Systems Research group in particular. Thank you all for the numerous amusing discussions during lunch, and for all the sailing trips, group dinners, and other social events. Some special thanks to the different roommates I had over the years: Vera, Joris, Mark, Savaş and Annerieke (in chronological order). I always found it very pleasant to have someone in the room to talk to, no matter whether it was about sense (such as new research directions) or nonsense (like sport results, or our favourite flavours of tea).

Third, I wish to thank all co-authors (other than Jan and Catholijn) who contributed to chapters of this thesis: Martine Delfos, Sander Los, Lourens van der Meij, Valentin Robu, Martijn Schut, and Leon van der Torre. Thank you very much for the pleasant cooperation

and for allowing me to include our papers in this thesis. In addition, I specifically want to mention my “predecessor” Wouter Wijngaards, who contributed to the development of an earlier version of the software environment that is used within this thesis. Moreover, I am very grateful to all members of my reading committee and Ph.D. committee: Cristiano Castelfranchi, Antonio Damasio, Frank van Harmelen, Pim Haselager, Sander Los, John-Jules Meyer, Walter Schaeken, and Ron Sun. Thank you all for spending some of your precious time on reading my thesis, and for your constructive and insightful comments.

Furthermore, I am very grateful to everybody who plays a role in my personal life, for allowing me to combine work and leisure in a harmonious way. In particular, many thanks to the best family members one could ever imagine: Joram, Mirte, Veronie and Harold. Thank you for creating the warm atmosphere in which I grew up, and for the all the pleasant dinners, holidays, and other activities we always do together. Also many thanks to Charlotte’s family, for always treating me like your own son, since the first day I met you. And of course many thanks to all my friends from school, sports, and whatever more, simply for being my friends in good and bad times.

And last, but certainly not least, dear Charlotte, thank you for your endless love and companionship, and for your patient attention to my never-ending stories about work. Having had the opportunity to talk about my research again and again has been the finishing touch for my understanding of it, and thus for the completion of this thesis.

Contents

PART I INTRODUCTION AND BASIC TECHNIQUES

CHAPTER 1

Introduction

Introduction

1. Cognitive Modelling

The human mind is undoubtedly one of the most complex entities that exist in our world. For centuries, researchers from multiple disciplines have been studying its functioning, varying from ancient philosophers like Plato to modern neuroscientists. Much progress has been made, for example due to the invention of the Magnetic Resonance Imaging (MRI) technology, which allows researchers to watch what part of the brain is active during a specific mind process. Nevertheless, in all these years only a fraction of the mysteries of the human mind has been unraveled. The collection of processes that are performed by the human mind is sometimes indicated by *cognition*, and the research area that is concerned with the understanding of cognition is known as *Cognitive Science*. Researchers with different backgrounds are working in this field, including linguists, anthropologists, psychologists, neuroscientists, philosophers, and researchers in Artificial Intelligence. Since the human mind involves many different aspects, there are also many ways to study it. A recent approach, which has become more effective since the rapid development of Computer Science, is *Cognitive Modelling*. In (Detje, Dörner, and Schaub, 2003), this approach is described as follows.

> *Cognitive Modeling is a method to study the human mind. Cognitive Modelers try to explain the structure and the processes of the human mind by building them. As this, Cognitive Modeling is "Synthetic Psychology". A model of human cognition should mirror human mental activities, human errors, slips and mistakes. Cognitive Modelers try to understand how the human memory works, how the human memory is structured to reflect reality, how the human memory is used for the organisation of behaviour. The scope of Cognitive Modeling is widened beyond cognition to more general and more complicated forms of psychological processes which include social, emotional and motivational factors.* (Detje, Dörner, and Schaub, 2003)

As can be seen in this description, the authors indicate a widening of the interpretation of the notion of cognition, including aspects such as emotion and motivational factors. Throughout this thesis this wider interpretation will be used, so that emotional and motivational aspects, but also notions such as consciousness are subsumed.

2. Benefits of Cognitive Modelling

The benefits of Cognitive Modelling can be formulated from two different perspectives. First, from a Cognitive Science perspective, Cognitive Modelling can be seen as useful to explore the nature of the human mind and human intelligence. This is in line with the above claim that "cognitive modelers try to explain the structure and the processes of the human mind by building them". Thus, when a certain aspect of human intelligence is investigated in enough detail to create a model of it, the experimental results of such a model can offer new insights to the disciplines that initiated the research, such as Psychology and Philosophy. For example, when psychologists establish a theory T that describes a specific reasoning pattern that humans use in certain circumstances, the cognitive modeller can try to make a (computer) model M of theory T. If, subsequently, this model M predicts roughly the behaviour described by T, however with some small deviations, the psychologists can use these predictions to create a refined (and hopefully more realistic) theory, T'. Second, modelling the human mind can offer a source of

inspiration to construct intelligent artifacts. This is often emphasised from an AI perspective. Luger and Stubblefield (2002) give the following definition of AI:

> *AI is the study of the mechanisms underlying intelligent behavior through the construction and evaluation of artifacts that enact those mechanisms.* (Luger and Stubblefield, 2002)

This definition shows that an important challenge within AI is building intelligent artifacts. This implies that the field of Cognitive Modelling can provide input for AI: when building intelligent artifacts, techniques can be used that are similar to the mechanisms that human beings use. Such techniques might be beneficial because artifacts that show some form of human-like intelligence have a variety of advantages over artifacts that do not. For example, they may be more efficient, more flexible, have a more natural interaction with humans, and their behaviour may be explained in more understandable terms. For a concrete example of an artifact that has all these advantages, one could think of an automated opponent in simulation-based training.

Throughout the thesis, the Cognitive Science perspective is central. Thus, the focus is on the formal analysis and modelling of cognitive processes (such as reasoning, classical conditioning, and the processes that yield consciousness) with as its main goal to explore the nature of these processes. Nevertheless, while doing this, also the Artificial Intelligence perspective is sometimes considered. This means that, while modelling cognitive processes, from time to time the question is asked how the results can be used to create intelligent artifacts. For example, in Part III about negotiation, the dynamics of human negotiation processes are analysed, and based on the results several suggestions are made to improve the performance of computer negotiators. However, in these cases the contribution from the AI perspective is limited to indicating how the results of an analysis can be used to create intelligent artifacts (e.g., by specifying (formal) requirements for the behaviour of such artifacts). In contrast, the actual implementation of such artifacts is beyond the scope of this thesis.

3. Research Goal

In this thesis a novel approach to Cognitive Modelling is introduced, which focuses on analysis and simulation of the *dynamics* of cognitive processes. Dynamic aspects play an important role in most cognitive processes. For example, within human reasoning processes such dynamic aspects are posing reasoning goals, making assumptions and evaluating assumptions. Likewise, in the domain of classical conditioning, examples are building up preparation strength and reacting to stimuli. As a consequence, such cognitive processes cannot be understood, justified or explained without taking into account these dynamic aspects. Therefore, the main *research goal* of this thesis is to introduce a novel approach for the analysis of the dynamics of cognitive processes, and to explore its applicability in a variety of cognitive domains.

4. Underlying Principles

The approach introduced in this thesis is based on a number of underlying principles, which are summarised below:

- The dynamics of a process reveals itself by different *states* evolving over time. A state at a given point in time is a conjunction of all aspects (or state properties) of the world that hold at that time. A *trace* is a time-indexed sequence of states. A trace can be viewed as a trajectory in the multi-dimensional space of possible states,

i.e., it is a specific instance of the process under analysis, see also (Port and Gelder, 1995).

- Processes can be modelled at different levels of *aggregation*: at a *local* level, and at *non-local* levels.
- At a local level, processes can be described in terms of their basic mechanisms. These mechanisms, which are described by what are called *local dynamic properties* within this thesis, concern the smallest considered steps within the conceptualisation of the process under analysis.
- In practice, local dynamic properties mostly describe the process from an *internal* perspective, i.e., taking mental states of the agents (and their mutual temporal or causal relations) into account. The type of mental states considered may vary. When expressing dynamic properties from a *realist* perspective (Kim, 1996), the mental states represent certain 'real' physical (e.g., neurological) states. When expressing dynamic properties from a *functionalist* perspective (Kim, 1996), the mental states do not necessarily exist in reality, but may merely be instruments to describe the process adequately (e.g., beliefs, desires and intentions, see Dennett, 1987).
- Often local dynamic properties can be expressed in an *executable* format, i.e., for each state they prescribe a unique future state. These properties have two advantages: they can be used directly for *simulation* of the process under analysis and they can be depicted *graphically* (like causal graphs or influence diagrams).
- At a non-local level, processes can be described in terms of their overall characteristics (called *global dynamic properties* within this thesis), or in terms of the characteristics of parts of the process (called *intermediate dynamic properties* within this thesis). Both types of properties usually consist of more complex relationships between states at different time points.
- In practice, global dynamic properties mostly describe the process from an *external* perspective, i.e., referring to an agent's externally observable behaviour instead of its mental states.
- To facilitate the gradual formalisation of a process, all dynamic properties can be expressed in different formats: in an *informal* format (using natural language), a *semi-formal* format (using more structured natural language), and a *formal* format (using a logical temporal language). An informal format is suited for communication with domain experts, a formal format is suited for processing by a computer, and a semi-formal format can be used to bridge the gap between the two.
- As a result of expressing dynamic properties at different levels of aggregation, certain *logical interlevel relationships* can be identified between the dynamic properties of different levels. These relationships may indicate, for example, that a number of local properties together imply an intermediate property, or that a number of intermediate properties together imply a global property.
- An important challenge in Cognitive Modelling is to *verify* whether the local properties used to describe the basic mechanisms of a process give rise to some global properties that are expected to hold for the process.
- Another important challenge in Cognitive Modelling is to *validate* all dynamic properties used to describe the process by comparing them with empirical data.

5. Modelling Approach

The above principles have formed the basis for the development of a novel modelling approach that has been explored and exploited within all the chapters of this thesis. The main element within the approach is a sorted temporal language, based on first-order predicate logic, called the *Temporal Trace Language* (TTL). This language is built on atoms that can refer to traces, time points, and state properties. Dynamic properties can be expressed in TTL as temporal statements using the standard logical connectives and quantification operators. On the basis of TTL, a simpler temporal language has been developed to specify simulation models in a declarative manner, called the *Language and Environment for Analysis of Dynamics by SimulaTiOn* (LEADSTO). This language enables to model direct temporal dependencies between two state properties in successive states. Thus, it can be used to express executable dynamic properties.

For both languages, a dedicated software environment has been constructed. With respect to the LEADSTO language, a software environment was built that supports (1) the formal specification of executable dynamic properties and (2) simulation on the basis of these properties. With respect to the TTL language, another software environment was built that supports (1) the formal specification of (non-local) dynamic properties and (2) automated checking of these properties against formal traces. To perform the checking, the software takes a formalised dynamic property and a set of traces as input, and determines automatically whether the property holds for the trace. Note that the traces can be generated either by a computer (e.g., using the LEADSTO simulation software) or by humans. As a consequence, it can also be used to compare computer models with empirical data. The LEADSTO language and its software environment are introduced in detail in Chapter 2 of this thesis. The TTL language and its software environment are explained briefly in Chapter 3.

6. Methodology

Based on the languages TTL and LEADSTO and their supporting software environments, a general methodology can be formulated for the analysis of the dynamics of (cognitive) processes, which consists of a number of important elements (not in any particular order):

a) Identification of **local dynamic properties** for basic mechanisms of the process under investigation (in informal format), possibly in joint cooperation with a domain expert or on the basis of empirical evidence.

b) Formalisation of these local properties in terms of **executable dynamic properties**, thereby creating an executable model of the dynamics of the process.

c) Simulation of the dynamics of the process, on the basis of the executable dynamic properties, thereby generating **simulation traces**.

d) Identification of relevant **non-local dynamic properties** that are expected to hold (or not hold) for the process under investigation (in informal format).

e) Formalisation of these non-local properties in terms of **global dynamic properties** or **intermediate dynamic properties**.

f) Establishment of **interlevel relations** between the dynamic properties at different levels via mathematical proof.

g) **Verification** of the global and intermediate dynamic properties against the simulation traces.

h) Performing human experiments in order to obtain **empirical traces**.

i) **Validation** of the global, intermediate and executable dynamic properties against the (formalised) empirical traces.

The different elements of the methodology are depicted in Figure 1. Here, the arrows indicate the different actions mentioned above, and the ovals indicate the input and output of the actions. Solid arrows should be interpreted as necessary, and dotted arrows as optional. For example, informal local properties can be identified by just observing what happens in the world, or by also explicitly analysing empirical traces.

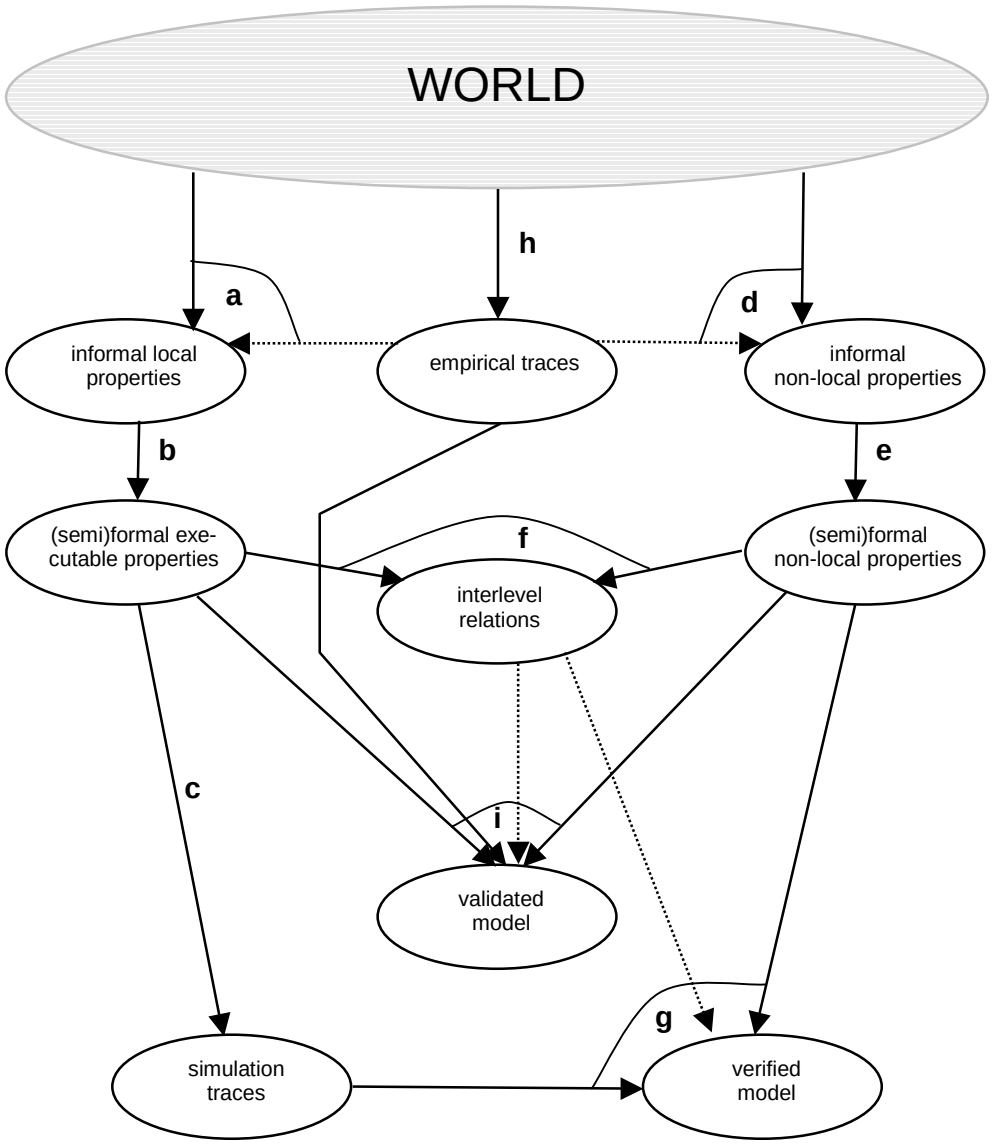


Figure 1. General methodology for the analysis of the dynamics of cognitive processes

Note that it is not always necessary to perform all actions as depicted in Figure 1. Instead, some research projects may focus on the analysis of empirical data, whereas others deal with the development and implementation of simulation models. This implies that, besides providing input for a next action, each oval in Figure 1 also has some added value in itself (or in combination with another element). For example, simulation traces serve as input for verification, but in the meantime they provide insight in the sequences of events

over time in specific instances of the process. Table 1 describes the separate benefits of the different elements of the methodology.

Entity	Benefit
informal local properties	insight in the basic mechanisms of the process (only after verification/validation)
(semi)formal executable properties	precise definition of the basic mechanisms of the process (only after verification/validation)
simulation traces	insight in the sequence of events over time in specific instances of the process
informal non-local properties	insight in the global dynamics of the process (only after verification/validation)
(semi)formal non-local properties	precise definition of the global dynamics of the process (only after verification/validation)
interlevel relations	formal theory of the dynamics of the process at different levels of aggregation, and their interdependencies
verified model	formally verified model of both the local and global dynamics of the process (and their interdependencies)
empirical traces	insight in the dynamics of empirical instances of the process (especially after formalisation of the traces)
validated model	empirically validated model of both the local and global dynamics of the process (and their interdependencies)

Table 1. Benefits of the different elements of the methodology

This methodology forms the backbone for (almost all of) the chapters in this thesis. The software environments mentioned earlier support several steps of the methodology. More specifically, the LEADSTO property editor can be used in b), the LEADSTO simulation tool can be used in c), the TTL property editor can be used in e), and the TTL property checker can be used in g) and i). In this thesis, action f) is performed by hand, in a way that is similar to proof methods in mathematics. This involves the establishment of logical relationships between the dynamic properties at different levels, in such a way that a number of dynamic properties at a certain level together imply a dynamic property at a higher level. However, work is currently in progress to develop automated support for this type of proofs as well.

Finally, note that, although action f) and g) might seem to have the same objective, there is an important difference. In g) it is checked whether non-local dynamic properties hold in a (limited) number of simulation traces, whereas in f) it is checked (implicitly) whether these properties hold for all possible traces of the simulation model. This means that the verification method used in f) is more exhaustive. However, a drawback of this method is that it may be more difficult to perform, since its complexity increases with the complexity of the properties and the size of the domain.

7. Related Work

The method introduced in this thesis is a novel approach to model the dynamics of cognitive processes. When comparing the method with existing approaches, it makes sense to consider two types of approaches:

1) approaches to model the dynamics of processes
2) approaches to model cognition

The first type of approaches is typically meant for modelling the temporal aspects of processes, but these processes are not necessarily cognitive processes. In contrast, the

second type of approaches is meant for modelling cognition, but these approaches do not focus explicitly on the dynamics of the cognitive processes. In principle, TTL and LEADSTO can be classified within the first group, since they are sufficiently generic to be used in a variety of non-cognitive domains. However, in this thesis it is explored to what extent the methods are appropriate to model cognitive processes in particular. Therefore, it is useful to compare them with existing cognitive architectures as well.

When compared to existing approaches for modelling dynamics of processes, an important advantage of both TTL and LEADSTO is that they allow the modeller to combine quantitative with qualitative aspects of the process under analysis. Traditionally, two classes of approaches to modelling dynamics are identified: *symbolic* modelling approaches, and *mathematical* modelling approaches, usually based on difference or differential equations. Symbolic approaches (also called logic-oriented approaches, see, e.g., Barringer *et al.*, 1996; Forbus, 1984) are good for expressing qualitative relations, but less suitable for working with quantitative relationships. Mathematical modelling approaches (e.g., Dynamical Systems Theory, see Port and Gelder, 1995), are good for the quantitative relations, but expressing conceptual, qualitative relationships is difficult. Nevertheless, the basic assumptions of both approaches are not fundamentally different. A main principle underlying Dynamical Systems Theory is the *state-determined system assumption* (Ashby, 1960), which assumes that the properties in subsequent states only depend on properties of the current state, not on properties of past states. Although this is often viewed as a specific feature of Dynamical Systems Theory, it was recently shown by Treur (2005) that the state-determined system assumption is unifyable with the assumptions underlying symbolic modelling approaches. These findings warrant the development of integrated approaches covering both qualitative and quantitative aspects. Proposals for such integrated approaches can already be found, for example, in (Sun, 2002). Also the languages put forward in this thesis are examples of such approaches. For example, the executable properties that are the basic building blocks of the LEADSTO language provide a useful representation format to specify state-determined systems in a logical manner, while still offering the possibility to express quantitative relations such as difference equations.

A number of architectures exist that are developed especially for modelling cognition, for example, ACT-R (Anderson and Lebiere, 1998), SOAR (Laird *et al.*, 1987), and COGENT (Cooper and Fox, 1998). As discussed above, these approaches are more specific than TTL and LEADSTO, since they focus on cognitive processes in particular. They basically consist of a number of different modules that reflect specific parts of cognition, such as memory, rule-based processes, and communication. Such cognitive architectures have in common with TTL and LEADSTO that they are hybrid approaches, supporting both qualitative and quantitative relations. However, in TTL and LEADSTO these qualitative and quantitative aspects can be combined within the same expressions, whereas in ACT-R, SOAR and COGENT separate modules exist to express them. Another exclusive feature of TTL and LEADSTO is the fact that these languages have a logical foundation.

8. Overview of the Thesis

The format of this thesis is a collection of articles. Most of the chapters are either reprints of refereed papers (which were published elsewhere), or extended versions of published papers. The papers are unchanged, with the exception of some layout-specific issues. This has two important implications. In the first place, there is overlap between a number of chapters. For example, each chapter contains a specific section in which the modelling

approach is introduced again, with special attention to the aspects of the approach that are relevant for the domain in question. Secondly, the fact that most chapters correspond to existing papers implies that each of them can be read in isolation. In other words, this thesis does not have any specific reading order. However, those readers that prefer to read the complete thesis are recommended to follow the normal order, starting with Chapter 1 and finishing with Chapter 21.

In this thesis, various cognitive processes in different (sub-)domains will be studied. The thesis has been structured according to six Parts, each focusing on a different aspect of cognitive processes:

I. *Introduction and Basic Techniques*
In Part I the topic of the thesis and the general research method are introduced. Chapter 1 (this chapter) introduces the topic of the thesis: a method for analysis of the dynamics of cognitive processes. Chapter 2 and 3 describe the main analysis techniques used throughout the thesis. Chapter 2 focuses on the LEADSTO language and software environment (for specification and formalisation of executable dynamic properties and simulation on the basis of these properties). Chapter 3 focuses on the TTL language and software environment (for specification and formalisation of complex dynamic properties and checking of these properties against traces).

II. *Reasoning*
Reasoning is a high-level cognitive function, and therefore generally considered as an important topic in Cognitive Science. In Part II, several reasoning patterns that are common in human reasoning are analysed using different techniques. Chapter 4 focuses on a specific reasoning pattern, called 'reasoning by assumption'. A case study is performed where participants are asked to solve a puzzle using this particular reasoning pattern. For the resulting empirical reasoning traces it is shown how they can be formalised and automatically analysed against various dynamic properties. In Chapter 5 a shift is made towards software agents. It is demonstrated how the type of dynamic properties identified in Chapter 4 can be used for requirements analysis of a software agent that performs reasoning by assumption. Next, Chapter 6 treats another reasoning pattern: 'multi-representational reasoning' and its control. In this type of reasoning, the states of the reasoning process may consist of different representations, such as arithmetical, geometrical and material representations. In this chapter a simulation model is presented and a formal analysis method for the dynamics of such reasoning processes is described. Finally, Chapter 7 addresses reasoning in a different context: reasoning within the domain of design of complex systems (e.g., software systems). The dynamics of this type of reasoning are simulated and analysed. It is described how important tasks in this type of reasoning are, among others, the identification of requirements and the assignment of components to the system to be designed.

III. *Negotiation*
Like reasoning, negotiation is a complex task that requires some intelligence. A difference with reasoning is that negotiation is performed by multiple agents together, whereas reasoning is restricted to a single agent. Thus, for negotiation also the dynamics of the interaction between agents is important, not only the dynamics of internal mental processes. In Part III, the dynamics of (human and automated) negotiation processes are analysed. The analysis comprises both a local perspective (focusing on the individual steps in the negotiation) and a global perspective (focusing on the overall welfare). Chapter 8 presents a generic software

environment for the analysis of closed multi-issue negotiation. It is shown how this environment can be used to check dynamic properties against negotiation traces. The traces may be generated by human negotiators, by software negotiators, or by both. Some preliminary results are provided of experiments in human multi-issue negotiation. To continue on this work, Chapter 9 reports on two experiments that contribute to the comparison of human- versus computer behaviour in multi-issue negotiation. Several advantages and disadvantages are shown of human strategies when compared to computer strategies. Based on the results of the experiments, several suggestions are made to improve the performance of computer negotiators.

IV. *Adaptivity*

Another characteristic of intelligent, cognitive agents is their ability to adapt to their experiences in a changing environment. For example, both humans and intelligent software agents may be able to learn from their mistakes. Part IV analyses the dynamics of two example processes where adaptivity plays an important role: human trace conditioning and adaptivity within psychotherapy. In Chapter 10 adaptivity within human trace conditioning is investigated: how do human beings learn to prepare for certain tasks? For this domain, an executable declarative logical model is created on the basis of Machado's mathematical model and a number of relevant global properties are verified against the simulated traces. Chapter 11 addresses another type of adaptivity: adaptivity of the human body in the case of eating regulation disorders. The dynamics of this process are simulated, both for wellfunctioning situations and for different types of malfunctioning situations that correspond to the first phase of well-known disorders such as anorexia (nervosa), obesitas, and bulimia. Moreover, these processes are analysed in terms of global dynamic properties and interlevel relations.

V. *Single vs. Multiple Agents*

In Part V it is shown how complex (cognitive) processes can be analysed from two different perspectives: a single agent and a multi-agent perspective. On the one hand, it is explored to what extent multi-agent processes that show some form of collective intelligence can be interpreted as a single agent. On the other hand, it is investigated how complex single-agent processes can be modelled as an organisation of multiple agents. Advantages of both perspectives are discussed. First, Chapter 12 introduces the principle of shared extended mind for multiple social agents: patterns created in the environment that the agents use as external mental states. This principle is illustrated by a case study of ant behaviour, of which the dynamics are formalised and simulated in terms of dynamic properties. Next, Chapter 13 addresses the question to what extent such a complex process involving multiple agents can be interpreted as a single agent process. It is shown for the example process of Chapter 12 how it can be conceptualised and formalised in two different manners: from a single agent or from a multi-agent perspective. Moreover, it is shown how an ontological mapping can be formally defined between the two formalisations, and how this mapping can be extended to a mapping of dynamic properties. Changing the perspective, Chapter 14 and 15 show how a complex single-agent process (within biology in this case) can be modelled as an organisation of multiple agents. The approach is illustrated in Chapter 14 for the case of the circulatory system in mammals, and in Chapter 15 for the case of the unicellular organism *E.coli*. In both chapters, different components of the system under analysis are modelled as roles in an organisation. Moreover, following the generic modelling approach used throughout this thesis, dynamic properties are

identified for different levels of the organisational model, and interlevel relations are made explicit.

VI. *Representational Content*
In Part VI the philosophical concept of 'representational content' for mental states is addressed. Assigning representational content to mental states is a fundamental challenge within AI, philosophy and cognition. Basically the question is here: 'what does it mean that an (artificial or real) agent has a mental state?'. For a number of mental states in different domains it is shown how their representational content can be defined in a precise manner, using formal expressions. In Chapter 16, first the concept of representational content is briefly introduced. After that, the applicability of a number of existing approaches to representational content is explored for a specific case study. This case study involves an example where an extensive interaction between agent and environment occurs, which is traditionally seen as a case where it is hard to impossible to define representational content. Next, Chapter 17 applies the idea of representational content to the neural mechanisms of classical conditioning. In this chapter, a simulation model is provided of the neural conditioning mechanisms of the sea hare Aplysia, which is one of the simplest (and therefore best understood) conditioning mechanisms in existing organisms. For a number of internal states of this model, it is shown how the representational content can be formally defined. Considering yet another domain, Chapter 18 applies the idea of representational content to the mental processes that lead to the birth of consciousness. In this chapter, a formal model is provided of Damasio's theory on core consciousness, and for a number of important concepts in the theory, the representational content is formally defined. Finally, Chapter 19 shows how the idea of representational content can be combined with the shared extended mind principle (see Chapter 12), thereby introducing a notion of collective representational content. Proposals for collective representational content are formalised and automatically verified.

VII. *Discussion and Future Work*
To conclude the thesis, in Part VII its main contributions are summarised and some future research possibilities are discussed. In Chapter 20 it is evaluated to what extent the research goal - of introducing and exploring the applicability of a novel approach for analysis of the dynamics of cognitive processes - has been reached. Several advantages and drawbacks of the approach are discussed. Finally, in Chapter 21 some possible directions for future work are indicated.

References

Anderson, J.R., and Lebiere, C. (1998). The atomic components of thought. Lawrence Erlbaum Associates, Mahwah, NJ.

Ashby, R. (1960). Design for a Brain. Second Edition. Chapman & Hall, London.

Barringer, H., M. Fisher, D. Gabbay, R. Owens, & M. Reynolds (1996). The Imperative Future:Principles of Executable Temporal Logic, Research Studies Press Ltd. and John Wiley & Sons.

Cooper, R., and Fox, J. (1998). COGENT: a visual design environment for cognitive modeling. Behavior Research Methods, Instruments & Computers, 30, pp. 553-564.

Dennett, D.C. (1987). The Intentional Stance. MIT Press, Cambridge, Massachusetts.

Detje, F., Dörner, D., and Schaub, H. (eds.) (2003). The Logic of Cognitive Systems: Proceedings of the Fifth International Conference on Cognitive Modeling, ICCM'03. Universitäts-Verlag Bamberg.

Forbus, K.D. (1984). Qualitative process theory. Artificial Intelligence, volume 24, number 1-3, pp. 85-168.
Gelder, T.J. van, and Port, R.F. (1995). It's About Time: An Overview of the Dynamical Approach to Cognition. In: (Port and Gelder, 1995), pp. 1-43.
Kim, J. (1996). Philosophy of Mind. Westview Press.
Laird, J.E., Newell, A., and Rosenbloom, P.S. (1987). Soar: an architecture for general intelligence. Artificial Intelligence, volume 33, issue 1, pp. 1-64.
Luger, G.F., and Stubblefield, W.A. (2002). Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Addison-Wesley, New York.
Port, R.F., and Gelder, T.J. van (eds.) (1995). Mind as Motion: Explorations in the Dynamics of Cognition. MIT Press, Cambridge, Mass.
Sun, R. (2002). Duality of the Mind. Lawrence Erlbaum Associates, Mahwah, NJ.
Treur, J. (2005). States of Change: Explaining Dynamics by Anticipatory State Properties. Philosophical Psychology Journal. In press.

PART I INTRODUCTION AND BASIC TECHNIQUES

CHAPTER 2

LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn

Part of this chapter appeared as Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. (2005). LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T., Kluegl, F., Lamersdorf, W., Klusch, M., and Huhns, M.N. (eds.), *Proceedings of the Third German Conference on Multi-Agent System Technologies, MATES'05*. Lecture Notes in Artificial Intelligence, vol. 3550, Springer Verlag, pp. 165-178.

A three-page abstract of this chapter appeared as Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. (2005). LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn (extended abstract). In: Ali, M. and Esposito, F. (eds.), *Proceedings of the 18th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, IEA/AIE 2005*. Lecture Notes in Artificial Intelligence, vol. 3533, Springer Verlag, pp. 363-366.

LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn

Tibor Bosse[1], Catholijn M. Jonker[2], Lourens van der Meij[1], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, lourens, treur}@cs.vu.nl
http://www.cs.vu.nl/~{tbosse, lourens, treur}
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.kun.nl

Abstract. This paper presents the language and software environment LEADSTO that has been developed to model and simulate dynamic processes in terms of both qualitative and quantitative concepts. The LEADSTO language is a declarative order-sorted temporal language, extended with quantitative means. Dynamic processes can be modelled by specifying the direct temporal dependencies between state properties in successive states. Based on the LEADSTO language, a software environment was developed that performs simulations of LEADSTO specifications, generates simulation traces for further analysis, and constructs visual representations of traces. The approach proved its value in a number of research projects in different domains.

1. Introduction

In simulations various formats are used to specify basic mechanisms or causal relations within a process, see e.g., [1], [6], [10]. Depending on the domain of application such basic mechanisms need to be formulated quantitatively or qualitatively. Usually, within a given application explicit boundaries can be given in which the mechanisms take effect. For example, “from the time of planting an avocado pit, it takes 4 to 6 weeks for a shoot to appear”.

In such examples, in order to simulate the process that takes place, it is important to model its *dynamics*. When considering current approaches to modelling dynamics, the following two classes can be identified: *logic-oriented* modelling approaches, and *mathematical* modelling approaches, usually based on difference or differential equations. Logic-oriented approaches are good for expressing qualitative relations, but less suitable for working with quantitative relationships. Mathematical modelling approaches (e.g., Dynamical Systems Theory [10]), are good for the quantitative relations, but expressing conceptual, qualitative relationships is very difficult. In this article, the LEADSTO language (and software environment) is proposed as a language combining the specification of qualitative and quantitative relations.

In Section 2, the LEADSTO language is introduced. Section 3 provides examples from existing case studies in which LEADSTO has been applied. Section 4 describes the tools that support the LEADSTO modelling environment in detail. In particular, the LEADSTO Property Editor and the LEADSTO Simulation Tool are discussed. Section 5 compares the approach to related modelling approaches, and Section 6 is a conclusion.

2. Modelling Dynamics in LEADSTO

Dynamics can be modelled in different forms. Based on the area within Mathematics called calculus, the Dynamical Systems Theory (DST) [10] advocates to model dynamics by continuous state variables and changes of their values over time, which is also assumed continuous. In particular, systems of differential or difference equations are used. This may work well in applications where the world states can be modelled in a quantitative manner by real-valued state variables and the world's dynamics shows continuous changes in these state variables that can be modelled by mathematical relationships between real-valued variables.

Not for all applications dynamics can be modelled in a quantitative manner as required for DST. Sometimes qualitative changes form an essential aspect of the dynamics of a process. For example, to model the dynamics of reasoning processes usually a quantitative approach will not work. In such processes states are characterised by qualitative state properties, and changes by transitions between such states. For such applications often qualitative, discrete modelling approaches are advocated, such as variants of modal temporal logic; e.g., [7]. However, using such non-quantitative methods, the more precise timing relations are lost too.

For the approach used in this paper, it was decided to consider time as continuous, described by real values, but to allow both quantitative and qualitative state properties. The approach subsumes approaches based on simulation of differential or difference equations, and discrete qualitative modelling approaches, but also combines them. For example, it is possible to model the exact (real-valued) time interval for which some qualitative property holds. Moreover, the relationships between states over time are described by either logical or mathematical means, or a combination thereof. This is explained below in more detail.

Dynamics is considered as evolution of states over time. The notion of state as used here is characterised on the basis of an ontology defining a set of properties that do or do not hold at a certain point in time. For a given (order-sorted predicate logic) ontology Ont, the propositional language signature consisting of all *state ground atoms* (or *atomic state properties*) based on Ont is denoted by APROP(Ont). The *state properties* based on a certain ontology Ont are formalised by the propositions that can be made (using conjunction, negation, disjunction, implication) from the ground atoms. A *state* S is an indication of which atomic state properties are true and which are false, i.e., a mapping S: APROP(Ont) $\rightarrow$ {true, false}.

To specify simulation models a temporal language has been developed. This language (the LEADSTO language) enables one to model direct temporal dependencies between two state properties in successive states, also called *dynamic properties*. A specification of dynamic properties in LEADSTO format has as advantages that it is executable and that it can often easily be depicted graphically. The format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the LEADSTO language the notation $\alpha \rightarrow\!\!\!\rightarrow_{e, f, g, h} \beta$ (also see Figure 1), means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

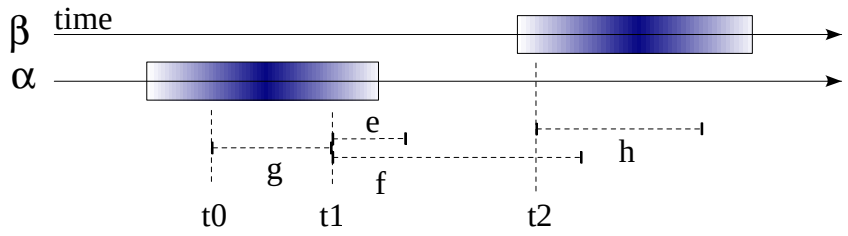


Figure 1. The timing relationships

An example dynamic property that uses the LEADSTO format defined above is the following: "observes(agent_A, food_present) $\rightarrow\!\!\!\!\rightarrow_{2,\ 3,\ 1,\ 1.5}$ belief(agent_A, food_present)". Informally, this example expresses the fact that, if agent A observes that food is present during 1 time unit, then after a delay between 2 and 3 time units, agent A will believe that food is present during 1.5 time units. In addition, within the LEADSTO language it is possible to use sorts, variables over sorts, real numbers, and mathematical operations, such as in "has_value(x, v) $\rightarrow\!\!\!\!\rightarrow_{e,\ f,\ g,\ h}$ has_value(x, v*0.25)".

Next, a *trace* or *trajectory* γ over a state ontology Ont is a time-indexed sequence of states over Ont (where the time frame is formalised by the real numbers). A LEADSTO expression $\alpha \rightarrow\!\!\!\!\rightarrow_{e,\ f,\ g,\ h} \beta$, holds for a trace γ if:

$\forall t1$: [$\forall t$ [$t1-g \leq t < t1 \Rightarrow \alpha$ holds in γ at time t] $\Rightarrow \exists d$ [$e \leq d \leq f$ & $\forall t'$ [$t1+d \leq t' < t1+d+h \Rightarrow \beta$ holds in γ at time t']

An important use of the LEADSTO language is as a specification language for simulation models. As indicated above, on the one hand LEADSTO expressions can be considered as logical expressions with a declarative, temporal semantics, showing what it means that they hold in a given trace. On the other hand they can be used to specify basic mechanisms of a process and to generate traces, similar to Executable Temporal Logic (cf. [1]).

Finally, the LEADSTO format can be graphically depicted in a causal graph-like format, by indicating state properties by circles and LEADSTO relationships by arrows, such as in Figure 2. The simple form leaves out the timing parameters e, f, g, h. A more detailed form can be obtained by placing the timing parameters in the picture as labels for the arrows. For more details about the LEADSTO language, see Section 4.

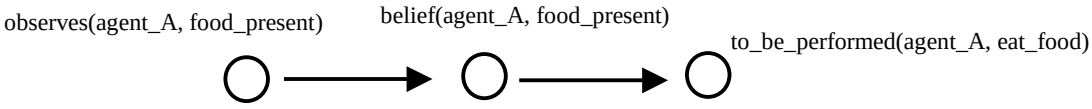


Figure 2. Example of a graphical representation of two LEADSTO properties

3. Applications

The LEADSTO environment has been applied in a number of research projects in different domains. In general, the research goal in these projects was to analyse the behavioural dynamics of agents in different domains. In most of them the focus was on cognitive processes, such as human reasoning, the creation of consciousness, and design tasks. LEADSTO was used to formalise the local dynamic properties of these processes at a high level of abstraction. Since they were specified in an executable format, such properties can be and have been used to generate simulation traces without additional

programming. In this section, for an example application, the formalisation of dynamic properties and the resulting simulation model will be discussed.

In [4], an adaptive dynamic model that describes normal functioning of eating regulation under varying metabolism levels is used as a basis for classification of eating regulation disorders, and of diagnosis and treatment within a therapy. Reasoning about the dynamic properties of this model (and disturbances of them) is performed in an intuitive, conceptual but informal manner. In [2], this model is formalised in LEADSTO, and some simulations are shown, both for wellfunctioning situations and for different types of malfunctioning situations that correspond to the first phase of well-known disorders such as anorexia (nervosa), obesitas, and bulimia. A number of LEADSTO expressions that have been used for the simulation are shown below:

LP1 (eat-stimulus)
The first local property LP1 expresses that an eat norm N and an intermediate amount eaten E less than this norm together lead to an eat stimulus. Formalisation:
intermediate_amount_eaten(E) and eat_norm(N) and E < N $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ stimulus(eat)

LP3 (increase of amount eaten)
Local property LP3 expresses how an eat stimulus increases an intermediate amount eaten by additional energy d (the energy value of what is eaten). Formalisation:
intermediate_amount_eaten(E) and stimulus(eat) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ intermediate_amount_eaten(E+d)

LP5 (day amount eaten)
Local property LP5 expresses that the day amount eaten is the intermediate amount eaten at the end of the day. Formalisation:
intermediate_amount_eaten(E) and time(24) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ day_amount_eaten(E)

LP6 (weight through balance of amount eaten and energy used)
Local property LP6 expresses a simple mechanism of how weight is affected by the day balance of amount eaten and energy used. Here γ is a fraction that specifies how energy leads to weight kilograms. Formalisation:
day_amount_eaten(E1) and day_used_energy(E2) and weight(W) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,25}$ weight(W + γ * (E1 – E2))

LP7 (adaptation of amount to be eaten)
Local property LP7 expresses a simple (logistic) mechanism for the adaptation of the eat norm based on the day amount of energy used. Here α is the adaptation speed, β is the fraction of E that is the limit of the adaptation; normally $\beta = 1$. Formalisation:
day_used_energy(E) and eat_norm(N) and time(24) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,25}$ eat_norm(N + α * N * (1 - N/βE))

LP9 (indication of anorexia)
Local property LP9 expresses that if the negative difference between the current weight and the recent weight is more than δ, then there is an indication that the patient has anorexia. Formalisation:
weight(W1) and recent_weight(W2) and W1-W2 > δ $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ indication(anorexia)

In Figure 3 an example of a resulting simulation trace is shown. This example illustrates the pattern of a person with anorexia. This example demonstrates the power of LEADSTO to combine (real-valued) quantitative concepts with (conceptual) qualitative concepts. The result is an easy to read (important for the communication with the domain expert), compact, and executable representation of an informal cognitive model of eating regulation.

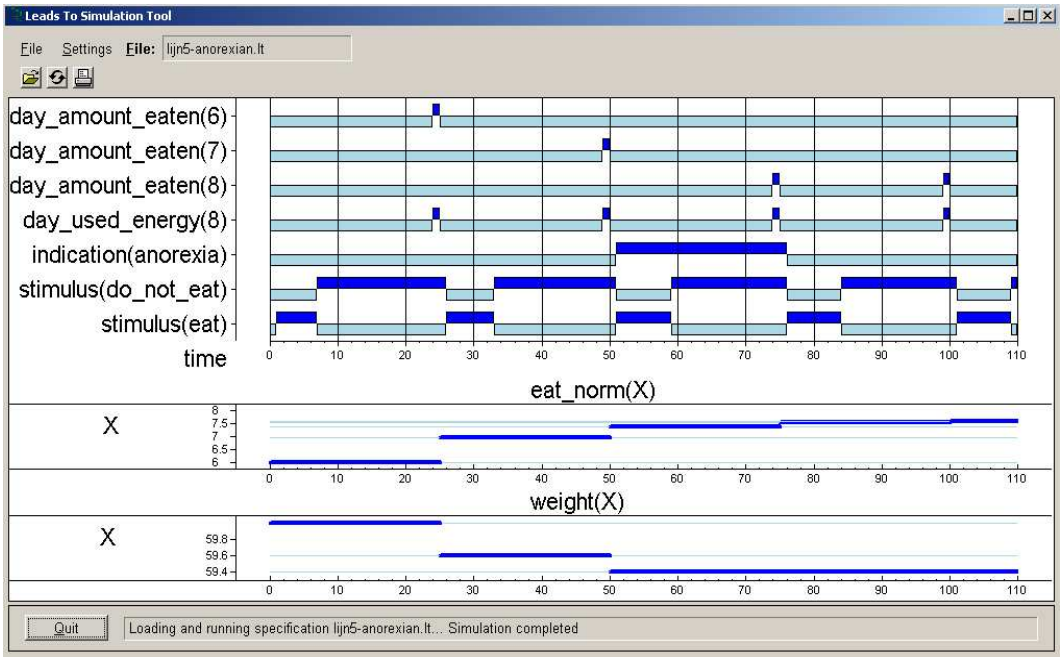


Figure 3. Example simulation trace

Another example application of LEADSTO can be found in [3]. In this paper, LEADSTO was used to model the dynamics of design processes. This example demonstrates the power of conceptual modelling based on highly abstract process descriptions. In less than 4 pages of code, the global dynamics of a design process are so well defined that the specification actually runs and designs a system. The specification took only a couple of days to construct, making the LEADSTO approach valuable for proof-of-concept simulations, thus important for Software Engineering.

4. Tools

In this section, the LEADSTO software environment is presented. Basically, this environment consists of two programs: the *Property Editor* (a graphical editor for constructing and editing LEADSTO specifications) and the *Simulation Tool* (for performing simulations of LEADSTO specifications, generating data-files containing traces for further analysis, and showing traces). Apart from the LEADSTO language constructs introduced in section 2 the LEADSTO software has a number of other language constructs. Section 4.1 discusses some details. Section 4.2 introduces the Property Editor. Section 4.3 deals with the Simulation Tool, and Section 4.4 describes the algorithm used to generate simulations.

4.1. Details of the LEADSTO language

There are various representations of LEADSTO specifications. A graphical representation is shown in Section 4.2 when discussing the Editor. In this section all language constructs are discussed using a formal representation, based on the way specifications are stored.

Variables. The language uses typed variables in various constructs. A variable is represented as <Var-Name>:<Sort>.

Sorts. Sorts may be defined as a set of instances that may be specified: sortdef(<Sort-Name>, [<Term>,…]). There are also built-in sorts such as integer, real, and ranges of integers represented as for example between(2,10).

Atoms. Atoms may be terms built up from names with argument lists where each argument must be a term or a variable, for example: belief(x:AGENT, food_present).

LEADSTO rules. LEADSTO rules are introduced in Section 2. They are represented as:
leadsto([<Vars>,] <Antecedent-Formula>, <Consequent-Formula>, <Delay>, where
<Delay> := efgh(<E-Range>,<F-Range>, <G-Range>,<H-Range>))[1]
<Vars> := "[" <Variable>,... "]"
For example, $\alpha \rightarrow\!\!\!\rightarrow_{0,\ 0,\ 1,\ 1} \beta$ is represented as leadsto(alfa, beta, efgh(0,0,1,1)). Variables occurring in LEADSTO rules must be explicitly declared as <Variable> entries.

Formulae. LEADSTO rules contain formulae. The current implementation allows conjunctions and universal quantification over typed variables. Some variables are global, encompassing the whole rule. Other - local - variables are part of universal quantification of some conjunction. The first kind of variables may be of infinite types. Currently, local variables must be of finite types. Some of these restrictions – such as on not allowing disjunction – will be removed in a next version. This will have no effect on the performance of the algorithm discussed in Section 4.4, but will make the details of the algorithm more complex. Other restrictions with respect to variables of infinite type will remain.

Time/Range. Time and Range values occurring in LEADSTO rules and interval constructs may be any number or expression evaluating to a number.

Constants. Constants may be defined using the following construct: constant(<Name>, <Value>). A constant(C1, a(1)) entry in a specification will lead to C1 being substituted by a(1) everywhere in the specification.

Intervals. During simulation, some atom values will be derived from LEADSTO rules. Others are not defined by rules but represent constant values of atoms over a certain time range. They are expressed as: interval([<Vars>,]<Range>,<LiteralConjunction>). Periodically reoccurring constant values are represented as:
periodic([<Vars>,]<Range>,<Period>,<LiteralConjunction>), where
<Range> := range(<Start-Time>,<End-Time>)
<Vars> := "[" <Variable>,... "]"
<Period> : an expression or constant or variable representing a number.
<LiteralConjunction> := <Literal> { and <Literal> }*
<Literal> := <Atom> | not <Atom>
For example, an entry interval([X:between(1,2)], range(10,20), a(X)) makes a(1) and a(2) true in the time range (10,20). Likewise, an entry periodic(P, range(0,1), 10) makes P true in time ranges (0,1), (10,11), (20,21), and so on.

Simulation range. The time range over which the simulation must be run is expressed by means of the constructs start_time(<Time>) and end_time(<Time>).

[1] The reason for grouping the delay is to make it easier to use delay constants.

Visualisation of Traces. The construct display(<Tag-Name>, <Property>) is used to specify details of how to display the traces. The <Tag-Name> argument makes it possible to define multiple views of a trace. The active view may be specified from within the User Interface of the Simulation Tool. A number of properties may be specified, for showing or hiding certain atoms, for sorting atoms, for grouping atoms into a graph, and so on.

4.2. Property Editor

The Property Editor provides a user-friendly way of building and editing LEADSTO specifications. It was designed in particular for laymen and students. The tool has been used successfully by students with no computer science background and by users with little computer experience. By means of graphical manipulation and filling in of forms a LEADSTO specification may be constructed. The end result is a saved LEADSTO specification file, containing entries discussed in section 4.1. Figure 4 gives an example of how LEADSTO specifications are presented and may be edited with the Property Editor.

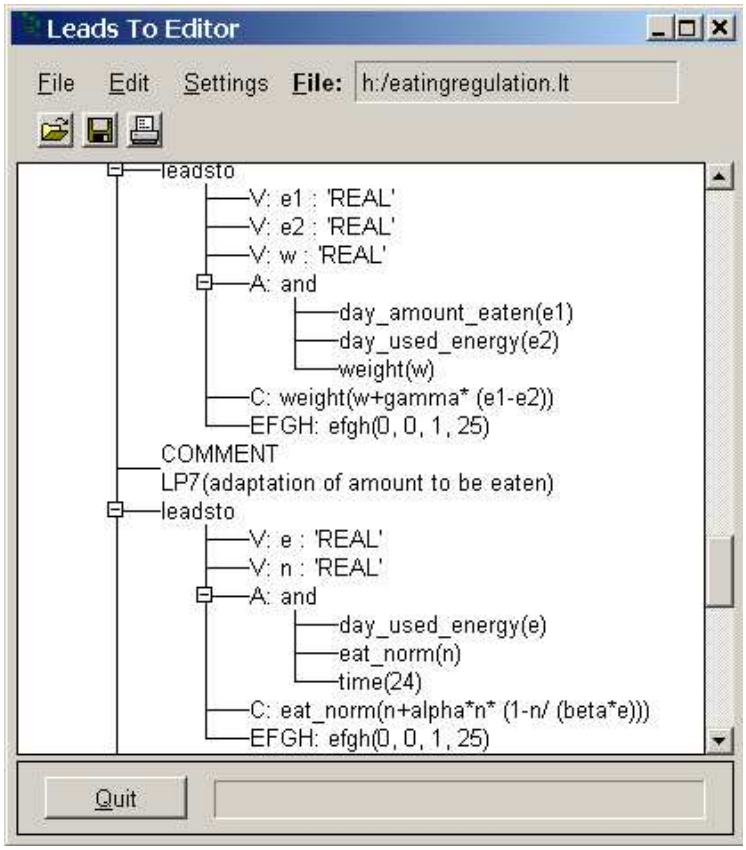


Figure 4. The LEADSTO Property Editor

4.3. Simulation Tool

Figure 5 gives an overview of the Simulation Tool and its interaction with the LEADSTO Property Editor. The bold rectangular borders define the separate tools. The lines with arrows represent data transport; the dashed arrows represent control. The Property Editor is used to generate and store LEADSTO specification files. The Simulation Tool loads these specification files. The overall control of the Simulation Tool is handled by the *Control-GUI* component. The Simulation Tool can perform the following activities:

- Loading LEADSTO specifications, performing a simulation and displaying the result.
- Loading and displaying existing traces (without performing simulation).
- Adjusting the visualisation of traces.

Loading and simulating a LEADSTO specification is handled in four steps:

1) The *Specification Loader* loads the specification.
2) The *Intermediate Code Generator* initialises the trace situation with values defined by interval and periodic entries in the specification. The LEADSTO rules are preprocessed: constants are substituted, universal quantifications are expanded and the rules are partially compiled into Prolog calls.
3) The actual simulation is performed by the *Runtime System*. This is the part that contains the algorithm, discussed in the next section.
4) At the end of a simulation the result is stored internally by the *Internal Trace Storage* component. The result can be saved as a trace file containing the evolution over time of truth values of all atoms occurring in the simulation, and will be visualised by the *Trace Visualisation GUI*. In principle, traces are three-valued, using the truth values true, false, and unknown. Saved trace files can be inspected later by the simulation tool and can be used by other tools, e.g., for automated analysis.

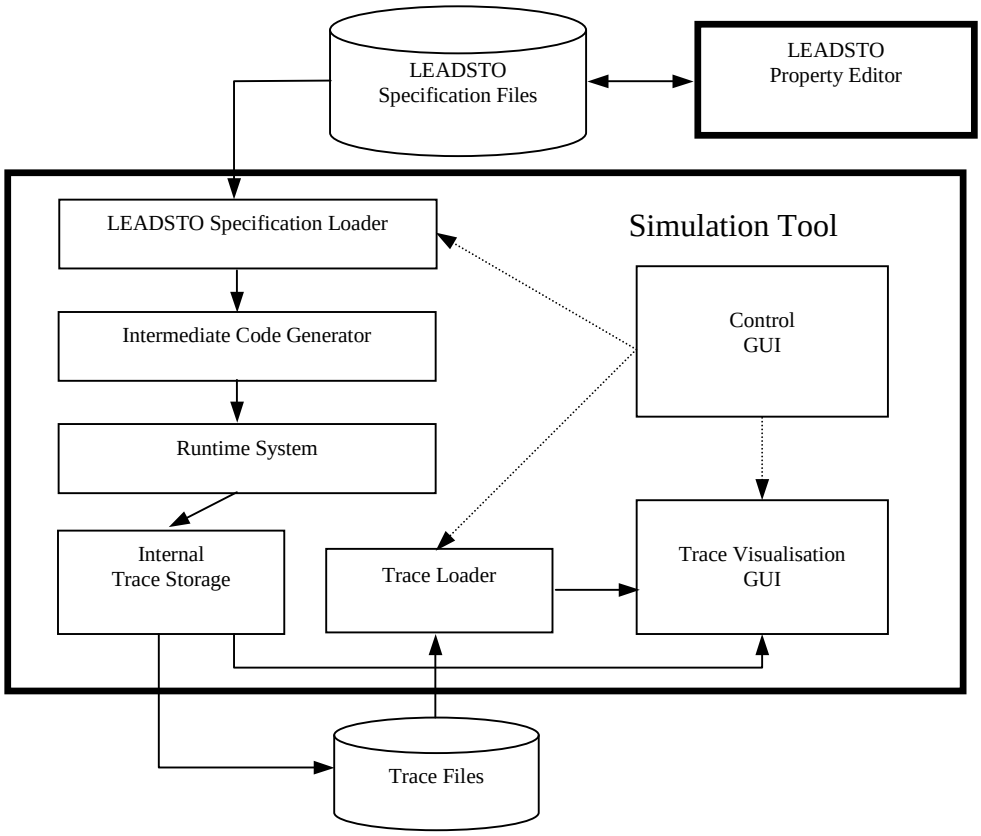


Figure 5. Simulation Tool Architecture

Note that visualisation of traces is integrated into the Simulation Tool through the *Trace Visualisation GUI* component. It is possible to select what atoms must be shown and in

what order (sorting) etc. Figure 3 is an example of the visualisation of the result of a simulation.

4.4. Simulation Engine Algorithm

In this section a sketch of the simulation algorithm is given. The core of the semantics is determined by the LEADSTO rules, for example leadsto(alpha,beta, efgh(e, f, g, h)) or (in the notation of Section 2) $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$. The state properties α, β are internally normalised. Currently, only state properties that can be simplified to conjunctions of literals are allowed.

Restrictions on delays. The parameters g and h are time intervals, they must be >= 0. The algorithm allows only causal rules, e,f >= 0. Allowing e,f < 0 would lead to non-causal behaviour (any trace situation could have an effect arbitrarily in the past) and an awkward simulation algorithm. The causal nature of the semantics of LEADSTO rules results in a straightforward algorithm: at each time point, a bound part of the past of the trace (the maximum of all g values of all rules) determines the values of a bound range of the future trace (the maximum of f + h over all LEADSTO rules).

Outline of the algorithm. First all interval and periodic entries are handled by setting the ranges of atoms according to their definition. Next, for the algorithm a time variable HandledTime is introduced: all LEADSTO rules with antecedent range up to HandledTime have fired. The idea is to propagate HandledTime until HandledTime >= EndTime[2] via the following steps:
At a certain HandledTime, a value for NextTime is calculated. This will be the first time in the future after HandledTime that firing of a LEADSTO rule with its g-interval (see Figure 1) extending past HandledTime may have effect in the form of some consequent atom set. The time increment will be at least as big as the minimum of e + h over all LEADSTO rules.
An (optional) Closed World Assumption is performed for all selected atoms in the range (HandledTime, NextTime), i.e., all unknown atoms in this range are made false.
All LEADSTO rules are applied for which the range of the antecedent ends before or overlaps with NextTime.
Set HandledTime := NextTime
Continue with step 1 until HandledTime >= EndTime

Implementation details. The complexity of the current algorithm is proportional to the number of LEADSTO rules in the specification, to the number of incremental time steps of the algorithm (which is at most equal to the length of the simulation divided by the minimum of e + h over all LEADSTO rules) and (at most) to the number of matching antecedent atoms per LEADSTO rule (limited by the number of atoms set during the simulation). A number of optimizations already improve the performance, such as only considering antecedent atoms that have matching values in the (HandledTime, NextTime) time range and not considering LEADSTO rules that have been tested to not fire until some time in the future.
The software was written in SWI-Prolog/XPCE, and consists of approximately 20000 lines of code. The approach for the design and implementation has been to first focus on a complete implementation that is easily adaptable, with acceptable performance for the

[2] EndTime is the time up to which the simulation should be run.

current users. For an impression of the performance: the simulation of Section 3 took two seconds on a regular Personal Computer. More complex LEADSTO simulations have been created that take about half an hour to run. For example: one simulation with 170 LEADSTO rules, 2000 time steps, with 15000 atoms set, took 45 minutes.

Future improvements. There is room for further performance improvement of the algorithm. One possible improvement is to increase the time increment NextTime – HandledTime introduced in the algorithm above. Global analysis of dependency of LEADSTO rules should improve the performance, for instance by trying to eliminate simple rules with small values of their e + h parameters. Furthermore, the LEADSTO language is being extended with constructs for probabilistic rules, and with constructs for systematically generating traces of LEADSTO specifications for a range of parameters.

5. Related Work

In the literature, a number of modelling approaches exist that have similarities to the approach discussed in this paper. Firstly, there is the family of approaches based on differential or difference equations (see, e.g., [10]). In these approaches, to simulate processes by mathematical means, difference equations are used, for example, of the form: $\Delta x = f(x)\, \Delta t$ or $x(t + \Delta t) = x(t) + f(x(t))\, \Delta t$. This can be modelled in the LEADSTO language as follows (where d is Δt): has_value(x, v) $\rightarrow\!\!\rightarrow_{d, d, d, d}$ has_value(x, v+f(v)*d). This shows how the LEADSTO modelling language subsumes modelling approaches based on difference equations. In addition to those approaches the LEADSTO language allows to express qualitative and logical aspects.

Another modelling approach, Executable Temporal Logic [1], is based on temporal logic formulae of the form $\varphi \,\&\, \chi \Rightarrow \psi$, where φ is a past formula, χ a present formula and ψ a future formula. In comparison to this format, the LEADSTO format is more expressive in the sense that it allows order-sorted logic for state properties, and allows one to express quantitative aspects. Moreover, the explicitly expressed timing parameters go beyond Executable Temporal Logic. On the other hand, within Executable Temporal Logic it is allowed to refer to different past states at different points in time, and thus to model more complex relationships over time. For the LEADSTO language the choice has been made to model only the basic mechanisms of a process (e.g., the direct causal relations), like in modelling approaches based on difference equations, and not to model the more complex mechanisms.

The Duration Calculus [11] is a modal logic for describing and reasoning about the real-time behaviour of dynamic systems, where states change over time and are represented by functions from time (reals) to the Boolean values (0 and 1). It is an extension of Interval Temporal Logic [8], but with continuous time, and uses integrated durations of states as interval temporal variables. Assuming finite variability of state functions (i.e., between any two time points only a finite number of state changes occurs), the axioms and rules of Duration Calculus constitute a complete logic (relative to Interval Temporal Logic). A number of interesting tools have been created around (subsets of) Duration Calculus, see, e.g., [9] for information on model checking duration calculus formulae. Duration Calculus itself is not directly used for creating executable models, but environments for executable code exist (e.g., PLC automata, see [5]) for which a semantics is given in Duration Calculus.

Another family of modelling approaches based on causal relations is the class of *qualitative reasoning* techniques (see, e.g., [6]). The main idea of these approaches is to

represent quantitative knowledge in terms of abstract, qualitative concepts. Like the LEADSTO language, qualitative reasoning can be used to perform simulation. A difference with LEADSTO is that it is a purely qualitative approach, and that it is less expressive with respect to temporal and quantitative aspects.

6. Conclusion

This article presents the language and software environment LEADSTO that has been developed especially to model and simulate dynamic processes in terms of both qualitative and quantitative concepts. It is, for example, possible to model differential and difference equations, and to combine those with discrete qualitative modelling approaches. Existing languages are either not accompanied by a software environment that allows simulation of the model, or do not allow the combination of both qualitative and quantitative concepts.

The language LEADSTO is a declarative order-sorted temporal language extended with quantitative notions (like integer, and real). Time is considered linear, continuous, described by real values. Dynamics can be modelled in LEADSTO as evolution of states over time, i.e., by modelling the direct temporal dependencies between state properties in successive states. The use of durations in these temporal properties facilitates the modelling of such temporal dependencies. Main advantages of the language are that it is executable and allows for graphical representation.

The problem with real time is that it is dense, i.e., between any two time points, a third time point can be identified. As a consequence, accurately modelling the dynamics of processes may require the use of a dense notion of time, instead of the more practiced variants of discrete time. The problem in a dense time frame of having an infinite number of time points between any two time points is tackled in LEADSTO by the assumption of "Finite Variability" (see Section 5 and, e.g., [11]).

The software environment LEADSTO was developed especially for the language. It features a dedicated Property Editor that proved its value for laymen, students and expert users. The core component is the Simulation Tool that performs simulations of LEADSTO specifications, generates simulation traces for further analysis, and visualises the traces.

The approach proved its value in a number of research projects in different domains. It has been used to analyse and simulate behavioural dynamics of agents in cognitive science (e.g., human reasoning, creation of consciousness, diagnosis of eating disorders), biology (e.g., cell decision processes, the dynamics of the heart), social science (e.g., organisation dynamics including organisational change, incident management), and artificial intelligence (e.g., design processes, ant colony behaviour). LEADSTO is so rich that it can be used to model phenomena from diverse perspectives. It has, for example, been used to model cognitive processes from a psychological/BDI perspective and from a physical/neurological perspective. For more publications about these applications, the reader is referred to the authors' homepages.

References

[1] Barringer, H., M. Fisher, D. Gabbay, R. Owens, & M. Reynolds (1996). The Imperative Future: Principles of Executable Temporal Logic, Research Studies Press Ltd. and John Wiley & Sons.

[2] Bosse, T., Delfos, M.F., Jonker, C.M., and Treur, J, Analysis of Adaptive Dynamical Systems for Eating Regulation Disorders. Proceedings of the 25th Annual Conference of the

Cognitive Science Society, CogSci 2003. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2003.

[3] Bosse, T., Jonker, C.M., and Treur, J., Analysis of Design Process Dynamics. In: R. Lopez de Mantaras, L. Saitta (eds.), Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'04, IOS Press, 2004, pp. 293-297. http://www.cs.vu.nl/~wai/Papers/ECAI04designdyn.pdf.

[4] Delfos, M.F. (2002). Lost Figure: Treatment of Anorexia, Bulimia and Obesitas (in Dutch). Swets and Zeitlinger Publishers, Lisse.

[5] Dierks, H. PLC-automata: A new class of implementable real-time automata. In M. Bertran and T. Rus, editors, Transformation-Based Reactive Systems Development (ARTS'97), volume 1231 of Lecture Notes in Computer Science, pages 111-125. Springer-Verlag, 1997.

[6] Forbus, K.D. (1984). Qualitative process theory. Artificial Intelligence, volume 24, number 1-3, pp. 85-168.

[7] Meyer, J.J.Ch., and Treur, J. (volume eds.), Agent-based Defeasible Control in Dynamic Environments. Series in Defeasible Reasoning and Uncertainty Management Systems (D. Gabbay and Ph. Smets, series eds.), vol. 7. Kluwer Academic Publishers, 2002.

[8] Moszkowski, B., and Manna, Z. Reasoning in Interval Temporal Logic. In Clarke, E., and Kozen, D., editors, Proceedings of the Workshop on Logics of Programs, volume 164 of LNCS, pages 371–382, Pittsburgh, PA, June 1983. Springer Verlag.

[9] Pandya, P.K., Model checking CTL[DC]. In: Proceedings of TACAS 2001, Genova, LNCS 2031, Springer-Verlag, April 2001. Also as Technical Report TCS-00-PKP-2, Tata Institute of Fundamental Research, Mumbai, 2000. http://www.tcs.tifr.res.in/~pandya/dcvalid103.html.

[10] Port, R.F., Gelder, T. van (eds.) (1995). Mind as Motion: Explorations in the Dynamics of Cognition. MIT Press, Cambridge, Mass.

[11] Zhou, C., Hoare, C.A.R., and Ravn, A.P. A Calculus of Durations, Information Processing Letter, 40, 5, pp. 269-276, 1991.

PART I INTRODUCTION AND BASIC TECHNIQUES

CHAPTER 3

A Temporal Trace Language for the Analysis of Dynamic Properties of Complex Processes

A Temporal Trace Language for the Formal Analysis of Dynamic Properties

Tibor Bosse[1], Catholijn M. Jonker[2], Lourens van der Meij[1], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, lourens, treur}@cs.vu.nl
http://www.cs.vu.nl/~{tbosse, lourens, treur}
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl

Abstract. Within many domains, among which biological, cognitive, and organisational areas, multiple interacting processes occur with dynamics that are hard to handle. Current approaches to analyse the dynamics of such processes, often based on differential equations, are not always successful. As an alternative to differential equations, this paper presents the predicate logical Temporal Trace Language (TTL) for the formal specification and analysis of dynamic properties. This language supports the specification of both qualitative and quantitative aspects, and therefore subsumes specification languages based on differential equations. A special software environment has been developed for TTL, featuring both a Property Editor for building and editing TTL properties and a Checking Tool that enables the formal verification of properties against a set of traces. TTL has a number of advantages, among which a high expressivity and the possibility to define sublanguages for simulation and verification of entailment relations. TTL proved its value in a number of projects within different domains.

1. Introduction

Within many domains, among which biological, cognitive, and organisational areas, multiple interacting processes occur with dynamics that are hard to handle. For example, modelling the dynamics of intracellular processes, cognitive processes, and organisational processes poses real challenges for biologists, cognitive scientists and organisation theorists. Currently, within the areas mentioned, differential equations are among the techniques most often used to address this challenge, with limited success. Within these disciplines it is felt that more abstract modelling techniques are required to cope with the complexity. This paper introduces the Temporal Trace Language (TTL) as such an abstract technique for the analysis of dynamic properties within complex domains.

In Section 2, the three areas as mentioned are first discussed in some more detail, and the type of challenges they pose are explained. Then, in Section 3 it will be discussed how directions of solutions can be explored for the type of challenges identified, leading to the introduction of TTL. Next, in Section 4, the basic concepts of the TTL language are introduced. In Section 5 it is shown how TTL can be used to express different kinds of dynamic properties. Moreover, it is shown how dynamic properties that are expressed in related languages can be translated into TTL. Section 6 provides examples from existing case studies in which TTL has been applied. Section 7 describes the tools that support the TTL modelling environment in detail. In particular, the TTL Property Editor and the TTL Checker Tool are discussed. Section 8 is a conclusion.

2. Background

As mentioned above, processes in the areas of Biology, Cognitive Science, and Organisation Theory usually have a very complex dynamics. Below, some examples are put forward to illustrate this. Moreover, it is shown why existing approaches to model such dynamics are not always sufficient.

2.1. Challenges in Dynamics in Different Domains

In the case of Biology, with the advent of post-genomic research, the interpretation of genetic data in terms of cellular behaviour is more crucial than ever. For the first time, integrative understanding of collective cell functioning can be obtained as a result of large-scale functional genomics and gene expression (microarray) data projects. However, a prerequisite is the ability to integrate and model the effects of many cellular aspects such as gene expression, metabolic fluxes, protein-protein interactions, cell signalling, and so on. In the area of modelling intracellular processes, the most widely used approach is based on differential equations, which are integrated numerically (Westerhoff, 2001). For some small unicellular organisms, a few isolated chemical pathways are understood in sufficient kinetic detail to obtain a description of their import and primary processing of nutrients; e.g., in *Escherichia coli* (Rohwer *et al.*, 2000), or yeast (Teusink *et al.*, 2000).

However, within this area this approach is felt to have serious limitations in tackling more large-scale cellular systems. First, hundreds or more reaction parameters are needed, for which reliable values are rarely available (Teusink *et al.*, 2000). This can seriously compromise the feasibility of the general model. Second, actual behaviour of intracellular pathways may be much less complex than is theoretically possible on the basis of the complexity of the chemical processes (e.g., Rotterdam *et al.*, 2002). At best, and only if all system parameters and internal connections are known and sufficiently tuned, the traditional approach delivers a computer replica of (part of) the living cell, which is nevertheless almost as remote from human understanding as the target system itself. This is because the modelling approach requires a description that is complete, inherently low-level, detailed and complex.

Within the area of Cognitive Science in recent years the dynamical perspective on cognitive phenomena has received much attention. Often the Dynamical Systems Theory (DST) is taken as a point of departure (e.g., Port and Gelder, 1995). In the study of cognitive phenomena this theory emphasises dynamics, and usually assumes that modelling dynamics of cognitive phenomena can be done effectively by mathematical techniques of algebraic, difference and differential equations. DST is able to model the temporal aspects of events taking place on a continuous time scale; e.g., recognition time, response time, and time involved in motor patterns and locomotion. Many convincing examples have illustrated the usefulness of DST; however, they often only address lower-level cognitive processes such as sensory or motor processing. Some examples of higher-level cognitive processes have been addressed as well; e.g., DST models for decision making in Decision Field Theory (Busemeyer and Townsend, 1993). Areas for which a quantitative approach based on DST offers less are the dynamics of higher-level processes with mainly a qualitative character, such as reasoning, complex task performance, and certain capabilities of language processing. For connectionist modelling, considered a subspecies of DST (Beer 1995), the same can be said.

Another limitation of existing approaches to model dynamics of cognitive phenomena is that they have difficulties with the specification of more complex adaptive processes. For example, a property such as ‘exercise improves skill’ is difficult to express using

existing techniques. Also in the domain of classical conditioning such adaptive properties occur. For example, in (Los and Heuvel, 2001), the following property is put forward to be relevant in conditioning processes: 'the conditioned response takes more time to build up and decay and its corresponding asymptotic value is lower when its corresponding critical moment is more remote from the warning signal.' Traditional approaches often have problems capturing such relative properties, because they involve the comparison of multiple alternatives for the history.

As far as the area of Organisation Theory is concerned, organisations play a central role in society (Mintzberg, 1979). Organisations can be seen as adaptive complex information processing systems of (boundedly) rational agents, and as tools for control; central questions are (Lomi and Larsen, 2001):

- from the first view: "given a set of assumptions about (different forms of) individual behaviour, how can the aggregate properties of a system be determined (or predicted) that are generated by the repeated interaction among those individual units?"
- from the second view: "given observable regularities in the behaviour of a composite system, which rules and procedures - if adopted by the individual units - induce and sustain these regularities?".

Both views and problems require techniques and tools that support the analysis of organisation behaviour. Literature on Organisation Theory is largely informal or semi-formal; see for example, (Mintzberg, 1979). The idea of using simulation as a formal technique to analyse organisational dynamics stems already from the 1950s, however, the computing power of computers then restricted its applicability. Although several results based on those simulations were frequently cited in the literature, the idea of simulation was largely ignored.

2.2. Recent Attempts

Given the limitations described above, in various domains there is a need for alternative approaches for the analysis of complex dynamic processes. Within Biology, to further the insight in the biological functioning of the cell, approaches that abstract from biochemical detail, while allowing higher-order integration of new data sources, might be helpful. One possible approach is to focus on conglomerates of biochemical processes, which act as functional units, such as "metabolic pathway", "catabolism", "transcriptome" or "regulon". Although at an early stage, recent studies exist where some of these concepts are defined formally (e.g., Rohwer *et al.*, 1996; Schilling *et al.*, 2000). The agent-based modelling strategies proposed in (Jonker, Snoep, Treur, Westerhoff and Wijngaards, 2002; Bosse *et al.*, 2005c), allow a more high-level functional perspective, where the cell effectively makes decisions regarding its internal dynamics and behaviour, given the environmental circumstances. Modelled using this high-level perspective, the behaviour of a model is less complex than that resulting from hundreds of differential equations.

For cognitive modelling, extension of the DST repertoire with techniques that cover qualitative aspects on a higher level of abstraction may prove more effective to model higher-level cognitive phenomena. Recently, a number of studies on the applicability of such alternative techniques for the dynamics of higher-level cognitive phenomena show promising results. Examples of such phenomena are complex tasks (Brazier, Jonker, Treur and Wijngaards, 2000; Cornelissen, Jonker and Treur, 2003), practical human reasoning (Jonker and Treur, 2003a), beliefs, desires and intentions (Jonker, Treur and Vries, 2002), and trust (Jonker, Schalken, Theeuwes and Treur, 2004). Moreover, such a dynamic

modelling approach proves useful to identify fundamental relationships between the dynamics of mental states and the dynamics of interaction with the environment (Jonker and Treur, 2003b).

Within Organisation Theory, recently formal and computational modelling techniques like simulation receive more attention. This change in interest is due to the fact that modellers today are more aware of recent developments within Organisation Theory involving concepts like organizational behaviour and adaptation, organizational embeddedness, organizational ecology, and competitive survival. Examples of this formalisation trend can be observed in books such as (Prietula, Gasser, and Carley, 1997; Lomi and Larsen, 2001), and in a recently created journal: Computational and Mathematical Organisation Theory (e.g., Moss *et al.*, 1998). Formalised work on dynamics of high-level organisation models in, e.g., (Ferber, Gutknecht, Jonker, Mueller, and Treur, 2002; Jonker, Treur and Wijngaards, 2002), is a first step in this direction.

3. Desiderata and Perspective

In Section 3.1 some desiderata are put forward for a suitable approach for modelling complex dynamic processes. These desiderata have resulted in a novel perspective for the development of such an approach, based on the idea of checking dynamic properties on practically given sets of traces. In the current paper this perspective is taken as departure for the creation of the language TTL. The perspective is explained in Section 3.2

3.1. Desiderata for a Dynamic Modelling Approach

As can be seen in the discussion about the different areas as given above, the demands for dynamic modelling approaches suitable for these areas are nontrivial. Such *desiderata for modelling languages* include:

(1) modelling at the right level of abstraction

(2) expressivity for logical relationships

(3) expressivity for quantitative relationships

(4) both discrete and continuous modelling

(5) difference and differential equations should be subsumed

(6) expressivity for dynamic properties of varying complexity, for example including adaptivity

Moreover, analysis techniques that would be desirable concern both the generation and formalisation of simulated and empirical trajectories or traces, as well as analysis of dynamic properties of such traces and relationships between such properties. Such *desiderata for analysis techniques* include:

(a) generating traces by simulation based on quantitative, continuous variables

(b) generating traces by simulation based on qualitative, logical notions

(c) formalisation of simulated or empirical traces

(d) analysis of properties of simulated traces

(e) analysis of properties of empirical traces

(f) analysis of relationships between (e.g., global and local) properties of traces

Taken together, the desiderata gathered above are not easy to fulfill. Sometimes they may even be considered mutually exclusive. On the one hand, high expressivity is desired, but on the other hand feasible analysis techniques are demanded. To make automated support for these analyses feasible, often the strategy is followed to limit the expressivity of the modelling language, thereby compromising on the first list of desiderata. For example, the expressivity is limited to difference and differential equations as in DST (excluding logical relationships, compromising at least (2)), or to propositional modal temporal logics (excluding numerical relationships, compromising at least (3), (5), (6)). In the former case calculus can be exploited to do simulation and analysis (Port and van Gelder, 1995), fulfilling (a) but not (b), (d), (e) and (f). In the latter case, for example, simulation can be based on a specific executable format (e.g., executable temporal logic (Barringer et al, 1996), fulfilling (b) but not (a) and (f)) and model checking techniques can be exploited for analysis of relationships between dynamic properties, fulfilling (d) to (f), e.g., (Clarke *et al.*, 2000; Manna and Pnueli, 1995; Stirling, 2001).

3.2. Perspective of this paper

Within the literature on analysis of properties (verification), much emphasis is put on computation of entailment relations. This essentially means checking properties on the set of all theoretically possible traces of a process. To make that feasible, expressivity of the language for these properties has to be sacrificed to a large extent. However, checking properties on a practically given set of traces (instead of all theoretically possible ones) is computationally much cheaper, and therefore the language for these properties can be more expressive. Such a set can consist of one or a number of traces, obtained empirically or by simulation. By limiting the desiderata by giving up (f), but still keeping (c) to (e), a much more expressive language for properties can be dealt with; the sorted predicate logic temporal trace language TTL described in this paper is an example of this.

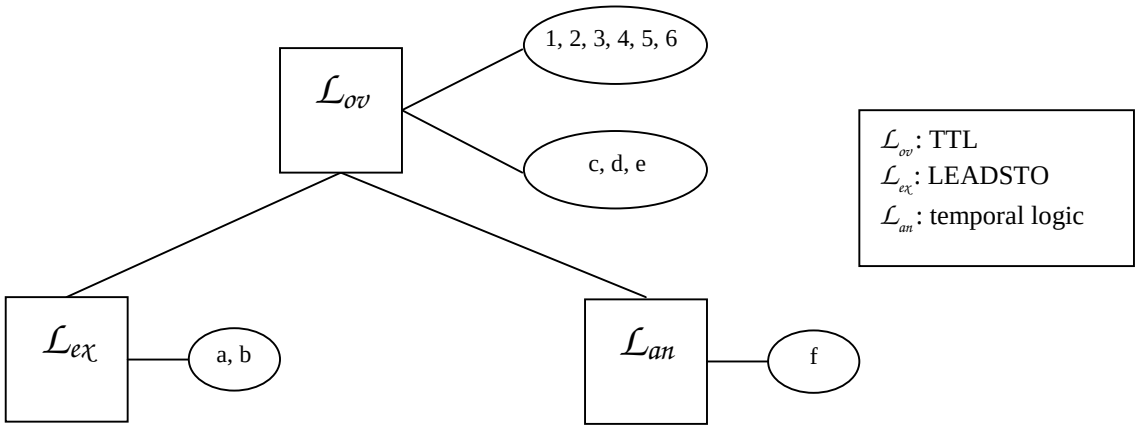


Figure 1. Embedding relationships between languages

For simulation it is essential to have limitations to the language. Therefore, an *executable language* can be defined as a sublanguage of the *overall language* for analysis. Moreover, also *analysis languages* that allow analysis in the sense of (f) can be embedded in the overall language. Thus the picture shown in Figure 1 is obtained. At the top there is an expressive overall language, in our case TTL, which fulfills all of the desiderata for modelling languages, i.e., (1) to (6). Concerning the desiderata for analysis techniques, it fulfills (c) to (e), but sacrifices (a), (b) and (f). In addition, a sublanguage can be defined

for execution (fulfilling (a) and (b)), and a sublanguage can be defined for analysis of relationships between properties in the sense of (f). For the case of TTL, one of the executable sublanguages that already exist is the LEADSTO language, cf. (Bosse *et al.*, 2005b). Moreover, for the sublanguage for analysis one could think of any standard temporal logic, such as LTL or CTL, see, e.g., (Benthem, 1983; Goldblatt, 1992). Having the language for simulation and the languages for analysis within one subsuming language provides the possibility to have a declarative specification of a simulation model, and thus to involve a simulation model in logical analyses.

The basic concepts of the language TTL are described in Section 4. In Section 5 it is shown how dynamic properties can be expressed in TTL, and how properties expressed in the LEADSTO language and in standard temporal logic can be translated into TTL, which shows these languages can be embedded in TTL and thus become sublanguages of TTL.

4. Basic Concepts

To describe dynamics, the notion of state is important. Dynamics will be described in the next section as evolution of states over time. The notion of state as used here is characterised on the basis of an ontology defining a set of physical and/or mental (state) properties (following, among others, (Kim, 1998)) that do or do not hold at a certain point in time. These properties are often called state properties to distinguish them from dynamic properties that relate different states over time. A specific state is characterised by dividing the set of state properties into those that hold, and those that do not hold in the state. Examples of state properties are 'the agent is hungry', 'the agent has pain', 'the agent's body temperature is 37.5° C', or 'the environmental temperature is 7° C'. Real value assignments to variables are also considered as possible state property descriptions. For example, in a DST approach based on variables x_1, x_2, x_3, x_4, that are related by differential equations over time, value assignments such as

$x_1 \leftarrow 0.06$
$x_2 \leftarrow 1.84$
$x_3 \leftarrow 3.36$
$x_4 \leftarrow -0.27$

are considered state descriptions. State properties are described by ontologies that define the concepts used.

4.1. Ontologies and State Properties

To formalise state property descriptions, ontologies are specified in a (many-sorted) first order logical format: an *ontology* is specified as a finite set of sorts, constants within these sorts, and relations and functions over these sorts (sometimes also called a signature). The example state properties mentioned above then can be defined by nullary predicates (or proposition symbols) such as hungry, or pain, or by using n-ary predicates (with $n \geq 1$) like has_temperature(body, 37.5), has_temperature(environment, 7), or has_value (x_1, 0.06).

For a given ontology Ont, the propositional language signature consisting of all *state ground atoms* based on Ont is denoted by At(Ont). The *state properties* based on a certain ontology Ont are formalised by the propositions that can be made (using conjunction, negation, disjunction, implication) from the ground atoms and constitute the set SPROP(Ont).

In many domains, it is desirable to distinguish different *agents* that are involved in the process under analysis. Moreover, it is often useful to distinguish between the internal,

external, input, and output state properties of these agents. To this end, the following different types of ontologies are introduced:

- IntOnt(A): to express *internal* state properties of the agent A
- InOnt(A): to express state properties of the *input* of agent A
- OutOnt(A): to express state properties of the *output* of the agent, and
- ExtOnt(A): to express state properties of the *external* world (for A)

For example, the state property pain may belong to IntOnt(A), whereas has_temperature(environment, 7), may belong to ExtOnt(A). The agent input ontology InOnt(A) defines properties for perception, the agent output ontology OutOnt(A) properties that indicate initiations of actions of A within the external world. The combination of InOnt(A) and OutOnt(A) is the *agent interaction ontology*, defined by InteractionOnt(A) = InOnt(A) ∪ OutOnt(A). The *overall ontology* for A is assumed to be the union of all ontologies mentioned above:

OvOnt(A) = InOnt(A) ∪ IntOnt(A) ∪ OutOnt(A) ∪ ExtOnt(A).

As yet no distinction between physical and mental internal state properties is made; the formal framework introduced in subsequent sections does not assume such a distinction. If no confusion is expected about the agent to which ontologies refer, the reference to A is sometimes left out.

4.2. Different Types of States

a) A *state* for a given ontology Ont is an assignment of truth-values {true, false} to the set of ground atoms At(Ont). The *set of all possible states* for an ontology Ont is denoted by STATES(Ont). In particular, STATES(OvOnt) denotes the set of all possible *overall states*. For the agent STATES(IntOnt) is the set of all of its possible *internal states*. Moreover, STATES(InteractionOnt) denotes the set of all *interaction states*.

b) The standard satisfaction relation ⊨ between states and state properties is used: S ⊨ p means that property p holds in state S. Here ⊨ is a predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus. For a property p expressed in Ont, the set of states over Ont in which p holds (i.e., the S with S ⊨ p) is denoted by STATES(Ont, p).

c) For a state S over ontology Ont with sub-ontology Ont', a restriction of S to Ont' can be made, denoted by S|Ont'; this restriction is the member of STATES(Ont') defined by S|Ont'(a) = S(a) if a ∈ At(Ont'). For example, if S is an overall state, i.e., a member of STATES(OvOnt), then the restriction of S to the internal atoms, S|IntOnt is an internal state, i.e., a member of STATES(IntOnt). The restriction operator serves as a form of projection of a combined state onto one of its parts.

5. Expressing Dynamic Properties

To describe the (internal and external) dynamics of an agent, explicit reference is made to time. Dynamic properties can be formulated that relate a state at one point in time to a state at another point in time. Some examples of dynamic properties of a certain agent are shown below, using an informal (natural language) notation.

A simple example is the following dynamic property specification for belief creation based on observation:

Observational belief creation
'At any point in time t1 if the agent observes at t1 that it is raining, then there exists a point in time t2 after t1 such that at t2 the agent believes that it is raining'.

Likewise, the persistence of a belief b over time can be specified by the dynamic property:

Belief persistence
'At any points in time t1 and t2 after t1, if the agent believes b at t1, then the agent will believe b at t2'.

An example of another type is trust monotonicity; this dynamic property specification about the dynamics of trust over time involves the comparison of two histories:

Trust monotonicity
'For any two possible histories, the better the agent's experiences with public transportation, the higher the agent's trust in public transportation'.

These examples were kept simple; they are just meant as illustrations. No attempt was made to make them as realistic as possible. As will be explained below, TTL can be used to express such dynamic properties, and other, more sophisticated ones, in a formal manner. First, in Section 5.1 the notion of trace is defined more explicitly. Next, in Section 5.2 it is shown in detail how dynamic properties can be expressed formally in TTL. After that, TTL will be compared with several related languages. In particular, in Section 5.3 it is shown how differential equations can be modelled in TTL. In Section 5.4 it is shown how executable properties expressed in LEADSTO can be translated into TTL, and in Section 5.5 it is shown how properties expressed in standard Linear Temporal Logic (LTL) can be translated into TTL.

5.1. Time Frame and Trace

a) A fixed time frame T is assumed which is linearly ordered. Depending on the application, it may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering.

b) A trace γ over an ontology Ont and time frame T is a time-indexed set of states

$$\gamma_t \ (t \in T)$$

in $STATES(Ont)$, i.e., a mapping

$$\gamma : T \rightarrow STATES(Ont).$$

For the specification of dynamic properties, these definitions work fine. However, for some specific operations (such as verification), a dense time frame may cause problems, since it consists of an infinite number of time points. Therefore, in such cases finite variability of state functions is assumed (i.e., between any two time points only a finite number of state changes occurs). This is discussed in more detail in Section 7.

The set of all traces over ontology Ont is denoted by $TRACES(Ont)$, i.e., $TRACES(Ont) = STATES(Ont)^T$.

c) A temporal domain description W is a given set of traces over the overall ontology, i.e.,

$$W \subseteq TRACES(OvOnt).$$

This set represents all possible developments over time (respecting the world's laws) of the part of the world considered in the application domain.

Different traces with respect to an agent A can refer to different experiments with A involving different worlds, or different events generated in the world. For human beings one can think of a set of experiments in cognitive science, in which different experiments are not assumed to influence the behaviour of the agent. For software agents, it is possible

to even erase the complete history (complete reset) and then activate the agent in a new world setting.

d) Given a trace γ over the overall ontology OvOnt, the input state of an agent A at time point t, i.e., γ_t |InOnt(A), is also denoted by

state(γ, t, input(A)).

Analogously,

state(γ, t, output(A))

denotes the output state of the agent at time point t,

state(γ, t, internal(A))

denotes the *internal state*, and

state(γ, t, external(A))

denotes the *external world state*. If no confusion is expected about the particular agent, the reference to A can be left out, e.g., as in state(γ, t, input). Moreover, the overall state of a system (agent and environment) at a certain moment, is denoted by state(γ, t).

5.2. Dynamic Properties

To express dynamic properties in a precise manner, it is needed to make explicit references to time points and traces. Comparable to the approach in situation calculus, TTL is built on atoms referring to, e.g., traces, time and state properties. For example, 'in the output state of A in trace γ at time t property p holds' is formalised by

state(γ, t, output(A)) $\models$ p.

Throughout the remainder of this paper, these kinds of atoms will be referred to as *Holds atoms*. Based on such Holds atoms, *Dynamic Properties* can be built using the usual logical connectives and quantification (for example, over traces, time and state properties). For example, the following dynamic properties can be expressed:

Observational belief creation
'In any trace, if at any point in time t1 the agent A observes that it is raining, then there exists a point in time t2 after t1 such that at t2 in the trace the agent A believes that it is raining'.

In formalised form:

$\forall \gamma \in W \;\; \forall t1$
[state(γ, t1, input(A)) $\models$ observation_result(itsraining)
$\Rightarrow \exists t2 \geq t1$ state(γ, t2, internal(A)) $\models$ belief(itsraining)]

Belief persistence
'In any trace, for any points in time t1 and t2 after t1, if the agent A has the belief b at t1 in the trace, then agent A has the belief b at t2 in this trace'.

In formalised form:

$\forall \gamma \in W \;\; \forall t1, t2$
[state(γ, t1, internal) $\models$ b & $t1 \leq t2$
$\Rightarrow$ state(γ, t2, internal) $\models$ b]

Trust monotonicity
For any two traces $\gamma 1$ and $\gamma 2$, if at each time point t the agent A's experience with public transportation in $\gamma 2$ at t is at least as good as A's experience with public transportation in $\gamma 1$ at t, then in trace $\gamma 2$ at each point in time t, the A's trust is at least as high as A's trust at t in trace $\gamma 1$'.

In formalised form:

∀γ1, γ2 ∈ W
[∀t [state(γ1, t, input(A)) ⊨ has_value(experience, v1) &
state(γ2, t, input(A)) ⊨ has_value(experience, v2)
⇒ v1≤ v2]
⇒
∀t [state(γ1, t, internal(A)) ⊨ has_value(trust, w1) &
state(γ2, t, internal(A)) ⊨ has_value(trust, w2)
⇒ w1≤ w2]]

Instead of the term Dynamic Property, sometimes the term *TTL Formula* is used within this paper. This is especially the case in Section 7, where the focus is on the technical aspects of the language.

5.3. Expressing Differential Equations in TTL

As mentioned in Section 2.1, especially in cognitive domains complex continuous relationships over time can be encountered. These relationships are often modelled semantically by differential equations, usually assumed to belong to the Dynamical Systems approach (DST), put forward, e.g., in (Port and Van Gelder, 1995). The question may arise whether or not such modelling techniques can be expressed in the Temporal Trace Language TTL. In this section it is shown how modelling techniques used in the dynamical systems approach, such as difference and differential equations, can be represented in TTL. First the discrete case is considered. An example of an application is the study of the use of logistic and other difference equations to model growth (and in particular growth spurts) of various cognitive phenomena, e.g., the growth of a child's lexicon between 10 and 17 months; cf. (Geert, 1991; 1995). The logistic difference equation used is:

$L(n+1) = L(n) (1 + r - r L(n)/K)$

Here r is the growth rate and K the carrying capacity. This equation can be expressed in our temporal trace language on the basis of a discrete time frame (e.g., the natural numbers) in a straightforward manner:

∀γ ∈ W ∀t
state(γ , t, internal) ⊨ has_value(L, v) ⇒
state(γ , t+1, internal) ⊨ has_value(L, v (1 + r - rv/K))

The traces γ satisfying the above dynamic property are the solutions of the difference equation. Another illustration is the dynamical model for decision-making presented in (Busemeyer and Townsend, 1993; Townsend and Busemeyer, 1995). The core of their decision model for the dynamics of the preference P for an action is based on the differential equation

$dP(t)/dt = -s P(t) + c V(t)$

where s and c are constants and V is a given evaluation function. One straightforward option is to use a discrete time frame and model a discretised version of this differential equation along the lines discussed above. However, it is also possible to use the dense time frame of the real numbers, and to express the differential equation directly. To this end, the following relation is introduced, expressing that $x = dy/dt$:

is_diff_of(γ, x, y) :
∀t,w ∀ε>0 ∃δ>0 ∀t',v,v'
0 < dist(t',t) < δ & state(γ , t, internal) ⊨ has_value(x, w)
& state(γ , t, internal) ⊨ has_value(y, v)
& state(γ , t', internal) ⊨ has_value(y, v')
⇒ dist((v'-v)/(t'-t),w) < ε

where dist(u,v) is defined as the absolute value of the difference, i.e. u-v if this is ≥ 0, and v-u otherwise. Using this, the differential equation can be expressed by:

is_diff_of(γ , - s P + c V, P)

The traces γ for which this statement is true are (or include) solutions for the differential equation.

Models consisting of combinations of difference or differential equations can be expressed in a similar manner. This shows how modelling constructs often used in DST can be expressed in TTL.

5.4. Expressing LEADSTO properties in TTL

As mentioned in Section 3.2, executable languages can be defined as sublanguages of TTL. An example of such a language, which was specifically designed for the simulation of dynamic processes in terms of both qualitative and quantitative concepts, is the LEADSTO language, cf. (Bosse *et al.*, 2005b). Below, it is shown how dynamic properties expressed in LEADSTO can be translated to TTL.

The LEADSTO language enables one to model direct temporal dependencies between two state properties in states at different points in time. A specification of dynamic properties in LEADSTO format has as advantages that it is executable and that it can often easily be depicted graphically. The format of LEADSTO is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the LEADSTO language the notation $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

In terms of TTL, the fact that the above statement holds for a trace γ can be expressed as follows:

∀t1[∀t [t1–g ≤ t < t1 ⇒ state(γ, t) ⊨ α] ⇒ ∃d [e ≤ d ≤ f & ∀t' [t1+d ≤ t' < t1+d+h ⇒ state(γ, t') ⊨ β]

5.5. Expressing Standard Temporal Logics in TTL

As mentioned in Section 3.2, besides executable languages also languages often used for the verification of entailment relations can be defined as sublanguages of TTL. Examples of such languages are LTL and CTL, see, e.g., (Benthem, 1983; Goldblatt, 1992). In this section, it is briefly shown how dynamic properties expressed as formulae in standard temporal logics can be translated to TTL; in particular, this will be illustrated for the case of LTL. The general idea is that this can be done in a rather straightforward manner by replacing the temporal operators of LTL by quantifiers over time. For example, consider the following LTL formula:

G(observation_result(itsraining) → F(belief(itsraining)))

where the temporal operator G means 'for all later time points', and F 'for some later time point'. The first operator can be translated into a universal quantifier, whereas the second one can be translated into an existential quantifier. Using TTL, this formula then can be expressed, for example, as follows:

∀t1 [state(γ, t1) ⊨ observation_result(itsraining) ⇒ ∃t2 ≥ t1 state(γ, t2) ⊨ belief(itsraining)]

However, note that the translation is not bi-directional, i.e., it is not always possible to translate TTL expressions into LTL expressions. An example of a TTL expression that cannot be translated to LTL is the property 'Trust Monotonicity' expressed in Section 5.2.

This property cannot be expressed in LTL since it involves the comparison of two different traces ($\gamma 1$ and $\gamma 2$ in this case). This shows that for example LTL can be considered a proper sublanguage of TTL, i.e., a sublanguage not equal to TTL. Similar observations can be made for other well-known temporal logics such as CTL.

To conclude, in Section 5.3 to 5.5 it was shown that languages such as DST, LEADSTO and LTL can be seen as sublanguages of the specification language TTL. Note that this does *not* imply that all *operations* that can be done using these languages (e.g., solving differential equations specified in DST, or performing simulation based on LEADSTO) can be performed using TTL tools. Each language has its own tools to perform specific operations. The tools that were specifically implemented for TTL will be introduced in Section 7.

6. Applications

The TTL language and its supporting software environment have been applied in a number of research projects in different domains. In general, the research goal in these projects was to analyse the behavioural dynamics of agents in different domains. In most of them the focus was on cognitive processes, such as human reasoning, the creation of consciousness, and design tasks. TTL was used to formalise dynamic properties of these processes at a high level of abstraction. Next, such properties were automatically checked against simulated or empirical traces. In this section, for an example application, the formalisation of dynamic properties will be discussed.

The example given in this section is taken from (Bosse *et al.*, 2005a). In that paper, the notion of representational content for mental states is discussed. Assigning representational content to mental states is a fundamental challenge within AI, philosophy and cognition. Basically, the question is here: 'what does it mean that an (artificial or real) agent has a mental state?'. In (Bosse *et al.*, 2005a), it is explored for a case study how representational content can be defined by means of so-called representation relations: expressions that relate an agent's mental state to an entire history (or future) of other states. Moreover, it is explored whether these representation relations can be formalised in terms of logical expressions.

For the case study, a domain was chosen that is generally viewed as a more complex case: a situation where the agent-environment interaction takes the form of an extensive reciprocal interplay in which both the agent and the environment contribute to the process in a mutual dependency. The specific example addressed involves the processes to unlock a front door that sticks. Between the moment that the door is reached and the moment that the door unlocks, the following reciprocal interaction takes place:

- the agent puts rotating pressure on the key
- the door lock generates resistance in the interplay
- the agent notices the resistance and increases the rotating pressure
- the door increases the resistance
- ...and so on, without any result...
- finally, after noticing the impasse, the agent changes the strategy by at the same time pulling the door and turning the key, which unlocks the door

To model this example, a specific domain ontology was created, consisting of Internal state properties, Input and Output state properties, and External state properties. This

ontology is given in Table 1. The left column denotes the predicates used, whereas the right column denotes the semantics of these predicates.

Internal state properties	
s1	sensory representation for being at the door
s2(r)	sensory representation for resistance r of the lock
p1(p)	preparation for the action to turn the key with rotating pressure p (without pulling the door)
p2	preparation for combined pulling the door and turning the key
c	state for having learnt that turning the key should be combined with pulling the door
Input state properties	
o1	observing being at the door
o2(r)	observing resistance r
Output state properties	
a1(p)	action turn the key with rotating pressure p (without pulling the door)
a2	action turn the key while pulling the door
External state properties	
arriving_at_door	the agent arrives at the door
lock_reaction(r)	the lock reacts with resistance r
door_unlocked	the door is unlocked
max_p(mp)	maximal force that can be exercised by the agent.
d(mr)	resistance threshold mr of the door (indicating that the door will continue to resist until pressure mr or more is used)

Table 1. Domain ontology used for the case study within (Bosse *et al.*, 2005a)

Based on this ontology, a simulation model has been created by formally specifying the dynamics between the state properties. Using the LEADSTO simulation software (Bosse *et al.*, 2005a), a number of simulation traces have been generated on the basis of this model. An example of such a trace is depicted in Figure 2.

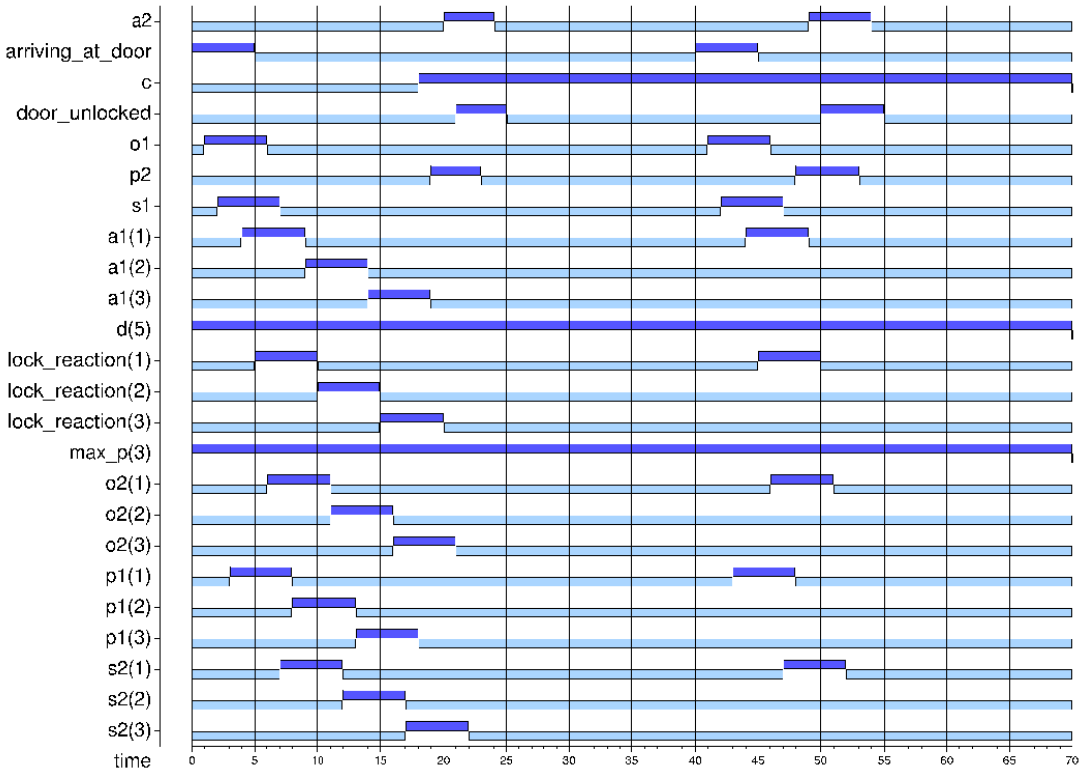


Figure 2. Example simulation trace for the case study within (Bosse *et al.*, 2005a)

Here, time is on the horizontal axis, the state properties are on the vertical axis. A dark box on top of the line indicates that the state property is true during that time period, and a lighter box below the line indicates that it is false.

After that, the next step was to specify representation relations for each of the internal state properties involved in the model (relating each internal state property to a history or future of other state properties, in such a way that the internal state property occurred in the simulation traces *if and only if* the sequence of other states occurred). TTL was used to formally specify these relations in terms of temporal-logical expressions. Some examples of these expressions are given below, both in semi-formal and in formal notation:

Representational Content of s1
'Internal state s1 occurs iff in the past input state o1 occurred'.
$\forall t1$ [state(γ, t1, input) $\models$ o1 $\Rightarrow \exists t2 \geq t1$ state(γ, t2, internal) $\models$ s1]
& $\forall t2$ [state(γ, t2, internal) $\models$ s1 $\Rightarrow \exists t1 \leq t2$ state(γ, t1, input) $\models$ o1]

Representational Content of p1(1)
'Internal state p1(1) occurs iff in the past input state o1 occurred'.
$\forall t1$ [state(γ, t1, input) $\models$ o1 $\Rightarrow \exists t2 \geq t1$ state(γ, t2, internal) $\models$ p1(1)]
& $\forall t2$ [state(γ, t2, internal) $\models$ p1(1) $\Rightarrow \exists t1 \leq t2$ state(γ, t1, input) $\models$ o1]

Representational Content of c
'Internal state c occurs iff in the past once o1 occurred, then a1(1), then o2(1), then a1(2), then o2(2), then a1(3), and finally o2(3)'.
$\forall t1,t2,t3,t4,t5,t6,t7$ [$t1 \leq t2 \leq t3 \leq t4 \leq t5 \leq t6 \leq t7$
& state(γ, t1, input) $\models$ o1
& state(γ, t2, output) $\models$ a1(1) & state(γ, t3, input) $\models$ o2(1)
& state(γ, t4, output) $\models$ a1(2) & state(γ, t5, input) $\models$ o2(2)
& state(γ, t6, output) $\models$ a1(3) & state(γ, t7, input) $\models$ o2(3)
$\Rightarrow \exists t8 \geq t7$ state(γ, t8, internal) $\models$ c]
& $\forall t8$ [state(γ, t8, internal) $\models$ c $\Rightarrow$
$\exists t1,t2,t3,t4,t5,t6,t7$ $t1 \leq t2 \leq t3 \leq t4 \leq t5 \leq t6 \leq t7 \leq t8$
& state(γ, t1, input) $\models$ o1
& state(γ, t2, output) $\models$ a1(1) & state(γ, t3, input) $\models$ o2(1)
& state(γ, t4, output) $\models$ a1(2) & state(γ, t5, input) $\models$ o2(2)
& state(γ, t6, output) $\models$ a1(3) & state(γ, t7, input) $\models$ o2(3)]

After formulating these expressions, it was desired to automatically check whether they were satisfied by the generated simulation traces. To be able to perform such checks (and similar checks in other domains), the TTL checker tool (see Section 7.2) was implemented. For the current case study, these checks all turned out to be successful, thereby validating (for the given traces at least) the choice for the representation relations.

Note that, in order to define appropriate representation relations in this domain, it was often needed to formulate complex temporal expressions, involving a large number of different time points (as for example in the last property given above). Due to the possibility of explicit reference to time points, TTL turned out to be well suitable for expressing such complex relations.

7. Tools

A specific software environment was built to support specification and verification of dynamic properties (represented as TTL formulae). Basically, this software environment consists of two closely integrated tools: the Property Editor and the Checker Tool. To explain how these tools work, Section 7.1 describes more details of the TTL language

from an implementation perspective. Next, Section 7.2 describes the actual operation of the tools. Finally, Section 7.3 discusses some implementation details of the Checker Tool.

7.1. Details of the TTL language

The previous sections introduced the TTL language in a somewhat informal way. However, the TTL software requires a strict representation. For instance, the implementation requires all variables in a TTL formula to be explicitly typed by specifying which sort they belong to. In this section, the TTL language is described in detail.

To enter TTL formulae in the correct format, the TTL Property Editor provides a graphical interface. The user fills in templates and builds up formulae by selecting building blocks from a menu. TTL specifications may also be supplied as plain text. Later in this section the syntax is given. First, some definitions are provided:

- A *TTL specification* consists of a number of user-defined property definitions and sort definitions. A property definition consists of a header (someprop(v1:s1, v2:s2), property name and formal arguments) and a body. The body is a TTL formula.
- A *TTL formula* is assembled from basic TTL formulae by conjunction, (Formula1 and Formula2), disjunction (Formula1 or Formula2), negation (not Formula), implication and quantification (forall ([v1:s1, v2:s2], Formula), exists ([v1:s1, v2:s2 < term2], Formula)).
- *Basic TTL formulae* are user-defined properties, Holds atoms, *predefined mathematical properties* (e.g. term1 = term2, term1 > term2) and built-in properties. The semantics of a user-defined property occurring in some TTL formula is one of substitution, not some kind of logic programming (recursion of properties is not allowed).
- *Holds atoms* are introduced in Section 5, e.g. state(trace1, t, output(ew)) $\models$ a1 $\wedge$ a2 .
- *Built-in properties* are complex properties encoded into the implementation language.
- TTL formula elements contain *terms* at various places: as restrictions on range variables, as actual parameter values in sub properties, within Holds atoms, and so on. Terms are "Prolog terms" (e.g., fn(t1,t2) , n1, t1 + t3, 1.3). Variables in terms are represented as X:sort1. Terms that are mathematical operations are evaluated, so the operands must be of an appropriate type. The functions begin(i:interval), end(i:interval), interval(t:time) and time(i:interval) introduced later are also terms that will be evaluated and substituted by their values.

For expressing more complex functions, the following building blocks are defined:

- case(Formula, Then, Else) where Formula is a TTL formula :
- f(case(Formula, Then, Else)) is equivalent to Formula and f(Then) or not Formula and f(Else).
- sum([v1:s1, v2:s2,..vn:sn], Term) where Term is a function of v1,..,vn: The sum of applying all tuples (v1,..vn) to Term.
- product(([v1:s1, v2:s2,..vn:sn], Term) where Term is a function of v1,..,vn: The product of applying all tuples (v1,..vn) to Term.

Furthermore, the language has a number of built-in *sorts* for integer, real and range of integers (sorts integer, real, between(i1:integer,i2:integer)). Sorts may be defined by enumerating their elements. There are predefined sorts for the set of all states (sort STATE) and the set of all loaded traces (sort TRACE, the temporal domain description set W introduced in section 5.1).

TTL formulae usually contain variables referring to time, specifically to time for a state property. In case a dense time frame is used, this may cause problems for the verification process, because an infinite number of time points must be considered. To deal with this problem, in the TTL tools *finite variability* of state functions is assumed. This assumption states that between any two time points only a finite number of state changes occurs. Thus, when a property is checked against a set of traces, the software determines time-intervals during which all atoms occurring in the property are constant in all traces. A built-in sort interval enumerates these disjoint time intervals. Values of this sort are ordered. A number of primitives are introduced to translate between interval values and time values:

- begin(i:interval) refers to the first time point of interval i.
- end(i:interval) refers to the last time point of interval i.
- interval(t:time) refers to the interval in which time point t occurs.
- time(i:interval) refers to a time point that occurs in interval i.

For an example in which one of these primitives is used, see the following Holds atom:

state(γ:TRACE, time(i:interval), internal) ⊨ a.

Moreover, libraries of predefined properties and functions are available, some generic, others for specific application domains; notably for the domain of multi-attribute negotiation there is a library of built-in properties available (Bosse *et al.*, 2005d).

An overview of the available language constructs in TTL is given in semi-formal syntax in Box 1. Here, the following remarks should be made:

- Bold text implies literal text,
- (..)* zero of more times whatever between parentheses,
- [..] optional
- <*ground-term*> : term that contains no <*variable*>
- The prefixes *sort-*, *variable-*, *var-*, *ont-*, *trace-*, and *time-* are only added for clarification and may be ignored. For example, <*trace-term*> can be read as < *term*>.
- holds(S, atom, true) is another notation for S ⊨ atom, holds(S, atom, false) for S ⊨ not atom.
- The last parameter <*ont-term*> of the <*state* > term representing the Ontology of the state must be input(A), output(A) or internal(A) (where A is an agent) or a variable with one of these values. The parameter can be omitted if there is no confusion about the Ontology.

```
<TTLformula>   : < TTLformula> <connector> <TTLformula>
               | ( <TTLformula> )
               | not <TTLformula>
               | <quantor> ( <qvars> , <TTLformula> )
               | holds( <state> , <atom> , <truthvalue> )
               | <state> ⊨ <sprop>
               | <property>
               | <term> <mathop> <term>
<sprop>                : <atom> | not <sprop> | <sprop> <connector> <sprop> | ( <sprop> )
<atom>                 : <compoundterm>
<compoundterm>         : <name> [ ( ( <term> , )* <term> ) ]
<term>                 : <variable> | <compoundterm> | <number>
                       | <term> <math-op> <term> | ( <term> )
<variable>             : <variable-name> : <sort-term>
<propertydefinition>   : define( <formalpropertyhead> , <TTL-formula> ).
<formalpropertyhead>   : <name> [ ( ( <formalvariable> , )* <formalvariable> ) ]
<formalvariable>       : <variable-name> : <sort-ground-term>
<connector>            : and | or | implies
<quantor>              : forall | exists
<state>                : state( <trace-term> , <time-term> )
                       | state( <trace-term>, <time-term>, <ont-term> )
<truthvalue>           : true | false
<mathop>               : '<' | '>' | '=' |
<sortdefinition>       : sortdef(<sort-name>,[ [( <ground-term> , )*<ground-term>]]).
<constant>             : <ground-term> = <ground-term> .
<qvars>                : [ [( <qvar> , )* <qvar> ] ]
<qvar>                 : [<term><mathop>]<var-name> :<sort-name> [<mathop><term>]
<TTLspecification>     : (<propertydefinition> | <sortdefinition> | <constant> )*
```

Box 1. Overview of TTL language constructs

7.2. Operation

As mentioned earlier, the TTL software environment comprises two closely integrated tools: the Property Editor and the Checker Tool. The Property Editor provides a user-friendly way of building and editing properties in the TTL language. It was designed in particular for less experienced users. By means of graphical manipulation and filling in of forms a TTL specification may be constructed (see Figure 3 for an impression). The Checker Tool can be used to check automatically whether a TTL formula holds for a set of traces. Operation of the tools involves three separate actions:

1) Loading, editing, and saving a set of TTL properties and user-defined sorts with the Property Editor (the big window in Figure 3).

2) Activating the Trace Manager (upper smaller window in Figure 3): loading and inspecting traces that will be checked and that will constitute the set of traces, the elements of sort TRACE (see section 7.1). Sources of traces can be both results of simulations such as output from the LEADSTO simulation software (see Bosse *et al.*, 2005b) and empirical traces such as traces of negotiations (see Bosse *et al.*, 2005d).

3) Selecting a menu entry "Check Property" while the cursor points to a property. The property is compiled (see Section 7.3 for details) and checked, and the result is presented to the user.

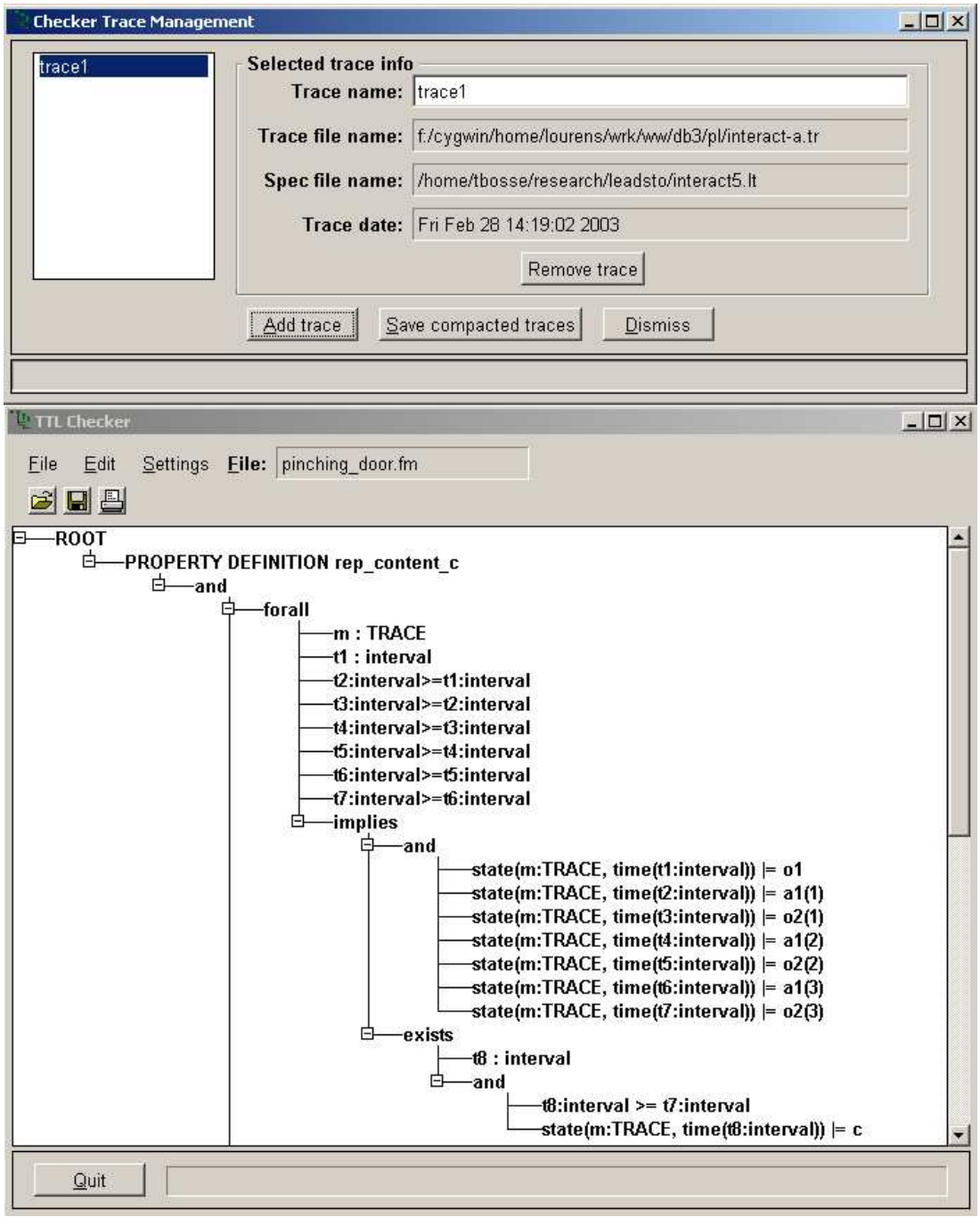


Figure 3. The TTL Checker with Trace Loader

In addition to the above, the TTL Checker has facilities for systematically loading traces and checking properties without user interaction. The software runs on Windows, Solaris and Linux platforms.

7.3. Implementation Details of the Checker

This section describes the algorithm used by the Checker Tool in detail. Fist, a number of introductory remarks are made:

- The Checker Tool was built specifically for the process of checking TTL formulae against traces. Here, a trace consists of a finite number of state atoms, changing a finite number of times. This has the following consequences:

- Using intervals instead of (continuous) time in TTL formulae will improve performance of the checking process (by simplifying quantification over time). Nevertheless, both options are possible.
- Other quantification variables will often refer to arguments of state atoms. There are a finite number of such state atoms. Iterating over values occurring in the traces will often be faster than iterating over all possible values of some variable.

- Checking may involve iteration over many values. Therefore, efficient coding is important. Compiling the formula that needs to be checked into code in the implementation language will improve performance (compared to interpretation).
- Checking may involve frequent access to values of state atoms. For acceptable performance, it is important to assure efficient access to state atoms specific to the formula that is checked.

The implementation is in Prolog (SWI-Prolog, the graphical user interface uses XPCE). A query to check some TTL formula against all loaded traces is compiled into a Prolog clause, which will succeed if the formula holds. The compilation proceeds as follows:

1) Fast access to state atoms is ensured: all atoms occurring in state properties within the TTL formula are gathered. Then, the set of all traces is analysed to determine the time intervals where all those atoms are constant. An index is built for fast access to all those atom values.
2) The TTL formula is compiled into Prolog: the formula is translated by mapping conjunction, disjunction and negation onto Prolog equivalents and by transforming universal quantification into existential quantification. For every variable occurring in the property, information about whether it is bound is maintained. If the first occurrence of some variable in a conjunction is in a Holds atom, then this variable becomes bound by code that binds the variable to successive matching Holds atoms; in a following element of the conjunction, the value may be used in expressions and evaluations in other members of the conjunction. If a variable is not bound by such an occurrence, but should be bound (because it appears in some mathematical operation or comparison), the variable must be bound by generating binding code to bind the variable to successive elements of the variable sort. If the sort is infinite, an error message is generated.

In addition to the above, some details are provided:

- It could be that in some conjunction a variable occurs both in the negation of some Holds atom and somewhere else in the conjunction. In that case the variable is marked "bound-lost", i.e., was bound but its value is lost through Prolog negation. Such a case would require the variable to be bound before the compiled code of the conjunction. The algorithm ensures that the coherence of variable instantiation is maintained and that whenever some variable occurs in a mathematical operation or comparison (compiled into Prolog equivalents), these variables are bound.
- State properties refer to atoms occurring in traces. These atoms may be unknown in some time interval. To avoid three-valued logic, state properties are transformed into disjunctive normal form and logical connectors are distributed outside the $\models$ relation. $S \models atom$ and $S \models not\ atom$ are translated into indexed access predicates to trace values.

To get an impression of the way the checker functions (without being complete), below a TTL formula is presented together with the resulting Prolog clause and checking result for some set of traces:

```
forall( [m:TRACE,t1 : interval]
        state(m, time(t1)) ⊨ o1
                implies
        exists([t2 : interval >= t1]
                state(m,time(t2)) ⊨ s1
    )
)
and
forall( [m:TRACE,t2 : interval]
        state(m, time(t2)) ⊨ s1
                implies
        exists([t1 : interval <= t2]
                state(m,time(t1)) ⊨ o1
    )
)
```

Notice that this formula corresponds to the dynamic property ‘Representational Content of s1’ presented in Section 6. Checking this formula against a set of traces produces the following output:

```
Checking s1_back...
Loading traces...
Compiling formula…
test :-
      \+((
         cholds:'holds(state(trace1, X), o1, true)|[]1'(A),
         between(0, 8, A),
         \+((
            cholds:'holds(state(trace1, X), s1, true)|[]1'(B),
            between(0, 8, B),
            B>=A
            ))
         )),
      \+((
         cholds:'holds(state(trace1, X), s1, true)|[]1'(C),
         between(0, 8, C),
         \+((
            cholds:'holds(state(trace1, X), o1, true)|[]1'(D),
            between(0, 8, D),
            D=<C
            ))
         )).

Formula s1_back satisfied
TIME:0 CPUTIME:00100144
```

[3]

In this output `test :- ....` is the listing of the compiled dynamic property. After that, the result of the check is printed out.

The specific optimizations discussed above make it possible to check realistic dynamic properties with reasonable performance. For an impression of the performance: checking the property ‘Representational Content of s1’ (see Section 6) against the trace depicted in that section took less than a second on a regular Personal Computer. Likewise, checking the property ‘Representational Content of c’ against that trace took about three minutes.

[3] cholds:’holds(…)..(A,B,..) are calls to optimally indexed trace values.

8. Conclusion

Within many domains, among which biological, cognitive, and organisational areas, multiple interacting processes occur with dynamics that are hard to handle. Current approaches to analyse the dynamics of such processes are often based on differential equations. However, for a number of applications these approaches have serious limitations. For example, in Biology, approaches based on differential equations have problems in tackling more large-scale cellular systems. Moreover, within Cognitive Science, such approaches are not particularly suitable to model higher-level processes with mainly a qualitative character, such as reasoning and complex task performance.

To deal with these limitations, this paper presents the predicate logical Temporal Trace Language (TTL) for the formal specification and analysis of dynamic properties. Although the language has a logical foundation, it supports the specification of both qualitative and quantitative aspects, and subsumes specification languages based on differential equations.

To support the formal specification and analysis of dynamic properties, a special software environment has been developed for TTL. This environment features both a dedicated Property Editor for building and editing TTL properties and a Checking Tool that enables the formal verification of properties against a set of traces, for example obtained from experiments or simulation. Although this form of checking is not as exhaustive as model checking (which essentially means checking properties on the set of all theoretically possible traces), in return, this makes it possible to specify more expressive properties. Furthermore, more specialised languages can be defined as a sublanguage of TTL. First, for the purpose of simulation, the executable language LEADSTO has been developed (Bosse *et al.*, 2005b). Second, for the verification of entailment relations, standard temporal languages such as LTL and CTL (see, e.g., (Benthem, 1983; Goldblatt, 1992)) can be defined as sublanguages of TTL.

As mentioned above, TTL has a high expressive power. For example, the possibility of explicit reference to *time points* and *time durations* enables modelling of the dynamics of continuous real-time phenomena, such as sensory and neural activity patterns in relation to mental properties, cf. (Port and van Gelder, 1995). Also difference and differential equations can be expressed. These features go beyond the expressive power available in standard linear or branching time temporal logics.

Furthermore, the possibility to quantify over traces allows for specification of *more complex adaptive behaviours*. As within most temporal logics, reactiveness and pro-activeness properties can be specified. In addition, in our language also properties involving different types of adaptive behaviour can be expressed. An example of such a property is ‘exercise improves skill’, which is a relative property in the sense that it involves the comparison of two alternatives for the history. Another property of this type is trust monotony: ‘the better the experiences with something or someone, the higher the trust’. Yet another example, in the domain of classical conditioning, is the following property: ‘the conditioned response takes more time to build up and decay and its corresponding asymptotic value is lower when its corresponding critical moment is more remote from the warning signal’ (Los and Heuvel, 2001). This type of relative property can be expressed in TTL, whereas in standard forms of temporal logic different alternative histories cannot be compared. Also, the kind of relative or comparative properties put forward in (Jackson and Pettit, 1990), such as ‘the more south on the northern hemisphere, the higher the trees’, as properties lacking an explanation in terms of a cause and its effects, can be expressed since our language allows comparison of different traces and different (local) restrictions within traces.

The possibility to define restrictions to *local languages for parts* of a system or the world is also an important feature. For example, the distinction between internal, external and input and output languages is crucial, and is supported by the language TTL, which also entails the possibility to quantify over system parts; this allows for specification of system modification over time.

Finally, since state properties are used as first class citizens in the temporal trace language, it is possible to explicitly refer to them, and to quantify over them, enabling the specification of what are sometimes called *second-order properties*, which are used in part of the philosophical literature (e.g., Kim, 1998) to express functional roles related to mental properties or states.

The approach discussed in this paper follows the standard view on calculus (based on epsilon-delta definitions). Recently, in (Gamboa, 2000; Gamboa and Kaufmann, 2001) an alternative approach, following the non-standard view (based on infinitesimals) has been presented for the integration of calculus within a logical (and theorem proving) framework. It may be the case, as claimed by some researchers, that for computational purposes the non-standard view has advantages. This will be an issue for further research.

To conclude, the approach proved its value in a number of research projects in different domains. It has been used to analyse behavioural dynamics of agents in cognitive science (e.g., human reasoning, creation of consciousness, diagnosis of eating disorders), biology (e.g., cell decision processes, the dynamics of the heart), social science (e.g., organisation dynamics including organisational change, incident management), and artificial intelligence (e.g., design processes, ant colony behaviour). For more publications about these applications, the reader is referred to the authors' homepages.

References

Barringer, H., M. Fisher, D. Gabbay, R. Owens, & M. Reynolds (1996). *The Imperative Future: Principles of Executable Temporal Logic*, Research Studies Press Ltd. and John Wiley & Sons.

Beer, R.D. (1995). Computational and dynamical languages for autonomous agents. In: (Port and van Gelder, 1995), pp. 121-148

Benthem, J.F.A.K., van (1983). The Logic of Time: A Model-theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse, Reidel, Dordrecht.

Bosse, T., Jonker, C.M., and Treur, J. (2005a). Representational Content and the Reciprocal Interplay of Agent and Environment. In: Leite, J., Omincini, A., Torroni, P., and Yolum, P. (eds.), *Proceedings of the Second International Workshop on Declarative Agent Languages and Technologies, DALT'04*. Lecture Notes in Artificial Intelligence, vol. 3476. Springer Verlag, pp. 270-288.

Bosse, T., Jonker, C.M., Meij, L., van der, and Treur, J. (2005b). LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn (extended abstract). *Proc. of the 18th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, IEA/AIE 2005*. Lecture Notes in AI, Springer Verlag. In press.

Bosse, T., Jonker, C.M., and Treur, J. (2005c). Modelling the Dynamics of Intracellular Processes as an Organisation of Multiple Agents. In: *Proc. of the First International Workshop on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics, MAS*BIOMED'05*, to appear.

Bosse, T., Jonker, C.M., Meij, L. van der, Robu, V., and Treur, J. (2005d). A System for Analysis of Multi-Issue Negotiation. In: Calisti, M., Klusch, M., and Unland, R. (eds.), *Software Agent-Based Applications, Platforms and Development Kits*. Birkhaeuser Publishing Company, in press.

Brazier, F.M.T., Jonker, C.M., Treur, J., and Wijngaards, N.J.E., On the Use of Shared Task Models in Knowledge Acquisition, Strategic User Interaction and Clarification Agents. *International Journal of Human-Computer Studies*, vol. 52, 2000, pp. 77-110.

Busemeyer, J., and Townsend, J.T. (1993). Decision field theory: a dynamic-cognitive approach to decision making in an uncertain environment. Psychological Review, vol. 100, pp. 432-459.

Clarke, E.M., Grumberg, O., and Peled, D.A. (2000). *Model Checking*. MIT Press.

Cornelissen, F., Jonker, C.M., and Treur, J. (2003). Compositional Verification of Knowledge-Based Task Models and Problem Solving Methods. *Knowledge and Information Systems Journal*.

Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J. (2002). Organization Models and Behavioural Requirements Specification for Multi-Agent Systems. In: Y. Demazeau, F. Garijo (eds.), *Multi-Agent System Organisations. Proc. of the 10th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'01*.

Gamboa, R. (2000). Continuity and Differentiability in ACL2. In M. Kaufmann, P. Manolios, and J.S. Moore (eds.), Computer-Aided Reasoning: ACL2 Case Studies. Kluwer Academic Press.

Gamboa, R., and Kaufmann, M. (2001). Nonstandard Analysis in ACL2. Journal of Automated Reasoning, vol. 27, pp. 323-351.

Geert, P. van (1991). A dynamic systems model of cognitive and language growth. Psychological Review, vol. 98, pp. 3-56.

Geert, P. van (1995). Growth Dynamics in Development. In: (Port and van Gelder, 1995), pp. 101-120.

Goldblatt, R. (1992). *Logics of Time and Computation*, 2nd edition, CSLI Lecture Notes 7.

Jackson, F., and Pettit, P. (1990). Program Explanation: A General Perspective. *Analysis*, vol. 50, pp. 107-117.

Jonker, C.M., Schalken, J.J.P., Theeuwes, J., and Treur, J, Human Experiments in Trust Dynamics. In: Jensen, C., Poslad, S., and Dimitrakos, T. (eds.), *Trust Management, Proc. of the Second International Conference on Trust Management, iTrust 2004*. Lecture Notes in Computer Science, vol. 2995, Springer Verlag, pp. 206-220.

Jonker, C.M., Snoep, J.L., Treur, J., Westerhoff, H.V., and Wijngaards, W.C.A. (2002). Putting Intentions into Cell Biochemistry: An Artificial Intelligence Perspective. *Journal of Theoretical Biology*, vol. 214, pp. 105-134.

Jonker, C.M., and Treur, J. (2003a). Modelling the Dynamics of Reasoning Processes: Reasoning by Assumption. *Cognitive Systems Research Journal*, vol. 4, pp. 119-136.

Jonker, C.M., and Treur, J. (2003b). A Temporal-Interactivist Perspective on the Dynamics of Mental States. *Cognitive Systems Research Journal*, vol. 4, pp. 137-155.

Jonker, C.M., Treur, J., and Vries, W. de (2002). Temporal Analysis of the Dynamics of Beliefs, Desires, and Intentions. *Cognitive Science Quarterly*. Special Issue on Desires, Goals, Intentions, and Values: Computational Architectures.

Jonker, C.M., Treur, J., and Wijngaards, W.C.A. (2002). Temporal Languages for Simulation and Analysis of the Dynamics Within an Organisation. In: B. Dunin-Keplicz and E. Nawarecki (eds.), *From Theory to Practice in Multi-Agent Systems, Proc. of the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems, CEEMAS'01*, 2001. Lecture Notes in AI, vol. 2296, Springer Verlag, 2002.

Kim, J. (1998). Mind in a Physical world: an Essay on the Mind-Body Problem and Mental Causation. MIT Press, Cambridge, Mass.

Lomi, A., and Larsen, E.R. (2001). *Dynamics of Organizations: Computational Modeling and Organization Theories*, AAAI Press, Menlo Park.

Los, S.A. and van den Heuvel, C.E. (2001). Intentional and Unintentional Contributions to Nonspecific Preparation During Reaction Time Foreperiods. *Journal of Experimental Psychology: Human Perception and Performance*, vol. 27, pp. 370-386.

Manna, Z., and Pnueli, A. (1995). *Temporal Verification of Reactive Systems: Safety*. Springer Verlag.

Mintzberg, H. (1979). The Structuring of Organisations, Prentice Hall, Englewood Cliffs, N.J.

Moss, S., Gaylard, H., Wallis, S. & Edmonds, B. (1998). SDML: A Multi-Agent Language for Organizational Modelling, *Computational and Mathematical Organization Theory* 4, (1), 43-70.

Port, R.F., Gelder, T. van (eds.) (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, Mass.

Prietula, M., Gasser, L., Carley, K. (1997). *Simulating Organizations*. MIT Press.

Rohwer, J.M., Schuster, S., Westerhoff, H.V. (1996). How to recognize monofunctional units in a metabolic system. J Theor Biol., 179(3):213-28.

Rohwer, J.M., Meadow, N.D., Roseman, S., Westerhoff, H.V., Postma, P.W. (2000). Understanding glucose transport by the bacterial phosphoenolpyruvate:glycose phosphotransferase system on the basis of kinetic measurements in vitro. J Biol Chem., 275(45):34909-21.

Rotterdam, B.J. van, Crielaard, W., van Stokkum, I.H., Hellingwerf, K.J., Westerhoff H.V. (2002). Simplicity in complexity: the photosynthetic reaction center performs as a simple 0.2 V battery. FEBS Lett., 510(1-2):105-7.

Schilling, C.H., Letscher, D., and Palsson, B. Ø. (2000). Theory for the Systemic definition of Metabolic Pathways and their use in Interpreting Metabolic Function from a Pathway-Oriented Perspective. In: *Journal of Theoretical Biology*, Vol. 203, pp. 229-248.

Stirling , C. (2001). *Modal and Temporal Properties of Processes*. Springer Verlag.

Teusink, B., Passarge, J., Reijenga, C.A., Esgalhado, E., van der Weijden, C.C., Schepper, M., Walsh, M.C., Bakker, B.M., van Dam, K., Westerhoff, H.V., Snoep, J.L. (2000). Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes? Testing biochemistry. Eur J Biochem., 267(17):5313-29.

Townsend, J.T., and Busemeyer, J. (1995). Dynamic Representation in Decision Making. In: (Port and van Gelder, 1995), pp. 101-120.

Westerhoff, H.V. (2001) The silicon cell, not dead but live! Metab Eng. 2001; 3(3):207-10.

PART II REASONING

CHAPTER 4

Reasoning by Assumption: Formalisation and Analysis of Human Reasoning Traces

This chapter will appear as Bosse, T., Jonker, C.M., and Treur, J. (2005). Formalisation and Analysis of Reasoning by Assumption. *Cognitive Science Journal*.

Part of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2005). Reasoning by Assumption: Formalisation and Analysis of Human Reasoning Traces. In: Mira, J. and Alvarez, J.R. (eds.), *Proceedings of the First International Work-conference on the Interplay between Natural and Artificial Computation, IWINAC 2005*. Lecture Notes in Artificial Intelligence, vol. 3561, Springer Verlag, pp. 427-436.

Part of a preliminary version of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2003). Reasoning by Assumption: Formalisation and Analysis of Human Reasoning Traces. In: Sun, R. (ed.), *Proceedings of the IJCAI 2003 Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions*, pp. 124-129.

Reasoning by Assumption: Formalisation and Analysis of Human Reasoning Traces

Tibor Bosse[1], Catholijn M. Jonker[2], Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR, Nijmegen, The Netherlands

Abstract. This paper introduces a novel approach for the analysis of the dynamics of reasoning processes, and explores its applicability for the reasoning pattern called 'reasoning by assumption'. More specifically, for a case study in the domain of a Master Mind game, it is shown how empirical human reasoning traces can be formalised and automatically analysed against dynamic properties they fulfil. To this end, for the pattern of 'reasoning by assumption' a variety of dynamic properties have been specified, some of which are considered characteristic for the reasoning pattern, whereas some other properties can be used to discriminate between different approaches to the reasoning. These properties have been automatically checked for the traces acquired in experiments undertaken. The approach turned out to be beneficial from two perspectives. First, checking characteristic properties contributes to the empirical validation of a theory on reasoning by assumption. Second, checking discriminating properties allows the analyst to identify different classes of human reasoners.

1. Introduction

Practical reasoning processes are often not limited to single reasoning steps, but extend to traces or trajectories of a number of interrelated reasoning steps over time. In the analysis of such reasoning processes, dynamic aspects play an important role. Examples of such dynamic aspects are posing reasoning goals, making assumptions and evaluating assumptions. As a consequence, such reasoning processes cannot be understood, justified or explained to others without taking into account these dynamic aspects. Therefore, the main goal of this paper is to present a novel approach for the analysis of the dynamics of reasoning processes. This approach is based on a combination of formal methods and human experiments. More specifically, it consists of a number of steps:

- First, a collection of *empirical data* is acquired, using an experiment in human reasoning.
- Next, the obtained transcripts are *formalised* using the Temporal Trace Language (TTL). This language was already shown to be a useful analysis tool for reasoning processes in (Jonker and Treur, 2002).
- Next, a number of *dynamic properties* of reasoning processes are formalised using TTL. These can be divided into two categories: *characterising properties* are expected to hold for all reasoning processes (e.g. 'the process terminates'), whereas *discriminating properties* are expected to hold for some reasoning processes (e.g., 'this particular reasoner uses the "stepwise" strategy').

- After that, using an *automated checking* tool, it is investigated which dynamic properties hold for which transcripts. Such an analysis can be useful in two different ways. On the one hand, checking characterising properties contributes to the validation of a theory on reasoning. On the other hand, checking discriminating properties helps to distinguish several types of transcripts from each other, thereby obtaining a classification of different reasoning strategies.
- Finally, *logical relationships* are established between different dynamic properties, indicating how a number of properties together entail another (global) property. As will be explained in Section 7, such logical relationships play an important role in the analysis of empirical reasoning processes.

A more detailed description of the different steps of the approach will be given in the remainder of this paper.

As a second contribution, this paper will demonstrate how the analysis approach can be applied for a specific reasoning pattern in human problem solving called 'reasoning by assumption'. This practical reasoning pattern involves a number of interrelated reasoning steps, and uses in its reasoning states not only content information but also meta-information about the status of content information and about control. For this reasoning pattern human reasoning protocols have been acquired, analysed, formalised, checked on dynamic properties and compared.

To obtain a specific case study in reasoning by assumption, the game of Master Mind was selected. This is a two-player game of logic, which was invented in 1970-71 by Mordecai Meirowitz (Nelson, 2000). The goal of the game is to discover a secret code of three colored pegs, which can be obtained by making guesses and receiving information about the correctness of the guesses. Because of its protocol, the pattern of reasoning by assumption occurs frequently within this game. Therefore, the game of Master Mind (in a simplified version) will be the main case study within this paper.

Below, in Section 2 the underlying dynamic perspective on reasoning is discussed in some more detail. Based on this perspective, a specific model for the pattern 'reasoning by assumption' is presented, adopted from (Jonker and Treur, 2003). In Section 3, the temporal language TTL, used to express properties of reasoning processes, is introduced in detail. Next, in Section 4 it is shown how think-aloud protocols involving reasoning by assumption in the game of Master Mind can be formalised to reasoning traces. A number of the dynamic properties that have been identified for patterns of reasoning by assumption are shown in Section 5. For the acquired reasoning traces the identified dynamic properties have been (automatically) checked. The results of these checks are provided in Section 6. In Section 7, it is shown how logical relationships between dynamic properties at different abstraction levels can play a role in the analysis of empirical reasoning processes. Section 8 discusses the difference between human strategies and optimal strategies, and Section 9 is a conclusion. Appendix A contains the complete list of relevant dynamic properties that have been identified for the pattern of reasoning by assumption. Appendix B contains a number of additional logical relationships between dynamic properties at different abstraction levels. Appendix C contains two example human transcripts, and their formalisation.

2. The Dynamics of Reasoning

In history, formalisation of the cognitive capability to perform reasoning has been addressed from different areas and angles: Philosophy, Logic, Cognitive Science, Artificial Intelligence. Within Philosophy and Logic much emphasis has been put on the

results (conclusions) of a reasoning process, abstracting from the process by which such a result is found: when is a statement a valid conclusion, given a certain set of premises. Within Artificial Intelligence, much emphasis has been put on effective inference procedures to automate reasoning processes. The dynamics of such inference procedures usually is described in a procedural, algorithmic manner; dynamics are not described and analysed in a conceptual, declarative manner. Within Cognitive Science, reasoning is often addressed from within one of the two dominant streams[4]: the syntactic approach (based on inference rules applied to syntactic expressions, as common in the logic-based approach, e.g., (Braine and O'Brien, 1998; Rips, 1994)), or the semantic approach (based on construction of mental models); e.g., (Johnson-Laird, 1983; Johnson-Laird and Byrne, 1991; Yang and Johnson-Laird, 2000; Yang and Bringsjord, 2001; Schroyens, Schaeken, and d'Ydewalle, 2001). Especially this second approach provides a wider scope than the scope usually taken within logic. Formalisation and formal analysis of the dynamics within (any of) these approaches has not been developed in depth yet.

To understand a specific reasoning process, especially for practical reasoning in humans, the dynamics are important. In particular, for reasoning processes in natural contexts, dynamic aspects play an important role and have to be taken into account, such as dynamically posing goals for the reasoning, or making (additional) assumptions during the reasoning, thus using a dynamic set of premises within the reasoning process. Decisions made during the process, for example, on which reasoning goal to pursue, or which assumptions to make, are an inherent part of such a reasoning process. Such reasoning processes or their outcomes cannot be understood without taking into account these dynamic aspects.

The approach to the semantical formalisation of the dynamics of reasoning presented in this section is based on the concepts reasoning state, transitions between reasoning states, and reasoning traces: traces of reasoning states. Based on these concepts, in Section 2.4 a specific model for the pattern 'reasoning by assumption' is presented, adopted from (Jonker and Treur, 2003).

2.1. Reasoning State

A reasoning state formalises an intermediate state of a reasoning process. It may include information on different aspects of the reasoning process, such as content information or control information. Within a syntactical inference approach, a reasoning state includes the set of statements derived (or truth values of these statements) at a certain point in time. Within a semantical approach based on mental models, a reasoning state may includes a particular mental model constructed at some point in time, or a set of mental models representing the considered possibilities. However, also additional (meta-)information can be included in a reasoning state, such as control information indicating what is the focus or goal of the reasoning, or information on which statements have been assumed during the reasoning. Moreover, to be able to cover interaction between reasoning and the external world, also part of the state of the external world is included in a reasoning state. This can be used, for example, to model the presentation of a reasoning puzzle to a subject, or to model the subject's observations in the world. The set of all reasoning states is denoted by RS.

[4] Recently, it was proposed by (Stenning and van Lambalgen, 2005) to reformulate the traditional distinction between syntactic and semantic approaches in terms of a distinction between reasoning towards an interpretation and reasoning from an interpretation. See Section 9 for a discussion about this topic.

2.2. Transition of reasoning states

A transition of reasoning states, i.e., an element $< S, S' >$ of RS x RS, defines a step from one reasoning state to another reasoning state; this formalises one reasoning step. A *reasoning transition relation* is a set of these transitions, or a relation on RS x RS. Such a relation can be used to specify the allowed transitions within a specific type of reasoning. Within a syntactical approach, inference rules such as modus ponens typically define transitions between reasoning states. For example, if two statements

$$p\ ,\ p \rightarrow q$$

are included in a reasoning state, then by a modus ponens transition, a reasoning state can be created where, in addition, also

$$q$$

is included. Within a semantical approach a construction step of a mental model, after a previous mental model, defines a transition between reasoning states. For example, if knowledge 'if p then q' is available, represented in a mental state

$$[p],\ q$$

and in addition not-q is presented, then a transition may occur to a reasoning state consisting of a set of mental models

$$p,\ q\ ;\ \sim p,\ \sim q;\ \sim p,\ q$$

which represents the set of possibilities considered; a next transition may involve the selection of the possibility that fits not-q, leading to the reasoning state

$$\sim p,\ \sim q$$

2.3. Reasoning trace

Reasoning dynamics or reasoning behaviour is the result of successive transitions from one reasoning state to another. By applying transitions in succession, a time-indexed sequence of reasoning states γ_t ($t \in T$) is constructed, where T is the time frame used (e.g., the natural numbers). A reasoning trace, created in this way, is a sequence of reasoning states over time, i.e., an element of RS^T. Traces are sequences of reasoning states such that each pair of successive reasoning states in this trace forms an allowed transition, as has been defined under transitions. A trace formalises one specific line of reasoning. A set of reasoning traces is a declarative description of the semantics of the behaviour of a reasoning process; each reasoning trace can be seen as one of the alternatives for the behaviour.

2.4. Reasoning by assumption

The specific reasoning pattern used in this paper to illustrate the approach is 'reasoning by assumption'. This type of reasoning often occurs in practical reasoning; for example, in

- Diagnostic reasoning based on causal knowledge
- Everyday reasoning
- Reasoning based on natural deduction

An example of diagnostic reasoning by assumption in the context of a car that won't start is:

> *'Suppose the battery is empty, then the lights won't work. But if I try, the lights turn out to work. Therefore the battery is not empty.'*

Note that on the basis of the assumption that the battery is empty, and causal knowledge that without a functioning battery the lights will not burn, a prediction is made

on an observable world fact, namely that the lights will not burn. After this an observation is initiated which has a result (lights do burn) that contradicts the prediction. Based on this outcome the assumption is evaluated and, as a result, rejected.

An example of an everyday process of reasoning by assumption is:

'Suppose I do not take my umbrella with me. Then, if it starts raining at 5 pm, I will get wet, which I don't want. Therefore I better take my umbrella with me'.

Again, based on the assumption some prediction is made, this time using probabilistic knowledge that it may rain at 5 pm. The prediction is in conflict with the desire not to get wet. The assumption is evaluated and rejected.

Examples of reasoning by assumption in natural deduction are:

- *Reductio ad absurdum or method of indirect proof*

  After assuming A, I have derived a contradiction. Therefore I can derive not A.

- *Implication introduction*

  After assuming A, I have derived B. Therefore I can derive that A implies B.

- *Reasoning by cases*

  After assuming A, I have derived C. Also after assuming B, I derived C. Therefore I can derive C from A or B.

Notice that as a common pattern in all of the examples presented, it seems that first a reasoning state is entered in which some fact is *assumed*. Next (possibly after some intermediate steps) a reasoning state is reached where *consequences* of this assumption have been *predicted*. Moreover, in some cases *observations* can be performed obtaining additional information about the world to be included in a next reasoning state. Finally, a reasoning state is reached in which an *evaluation* has taken place, for example, resulting in rejection of the assumption; possibly in the next state the assumption actually is retracted, and further conclusions are added.

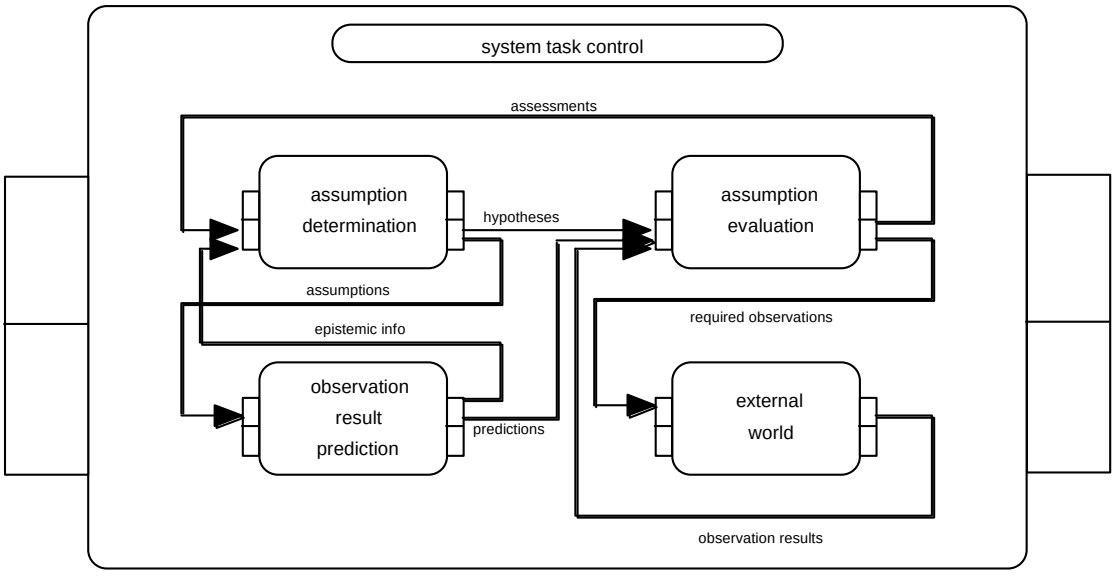


Figure 1. Model for Reasoning by Assumption

In (Jonker and Treur, 2003), this common pattern has been taken as a basis for the development of a (simulation) model for reasoning by assumption. According to this model, the process of reasoning by assumption involves three important sub-processes:

assumption determination, *observation result prediction*, and *assumption evaluation*. See Figure 1 for an overview of the model. In this figure, the rounded rectangles denote different components of the model where the different sub-processes take place (including the external world, which is used to observe the relevant predictions made). The arrows indicate information flow. Note that this model can be viewed as a refinement of Simon and Lea (1974)'s dual problem spaces model (see also Klahr and Dunbar, 1988), which distinguishes between a space for generation of hypotheses and a space for evaluation of these hypotheses. In the model depicted in Figure 1, an additional space is introduced for the prediction of the consequences of the hypotheses. In the original dual problem spaces model, this space was considered to be part of the space for hypothesis generation.

In the remainder of this paper, the above model for reasoning dynamics is taken as a point of departure in the formal analysis of human reasoning traces.

3. A Temporal Trace Language

In recent literature on Computer Science and Artificial Intelligence, temporal languages to specify dynamic properties of processes have been put forward; for example, (Dardenne, Lamsweerde and Fickas, 1993; Dubois, Du Bois and Zeipen, 1995; Herlea, Jonker, Treur, and Wijngaards, 1999). To specify properties on the dynamics of *reasoning* processes in particular, the temporal trace language TTL used in (Herlea *et al.*, 1999; Jonker and Treur, 1998) is adopted. This is a language in the family of languages to which also situation calculus (Reiter, 2001) and event calculus (Kowalski and Sergot, 1986) belong, and was also succesfully used to analyse multi-representational reasoning processes in (Jonker and Treur, 2002).

Ontology. An ontology is a specification (in order-sorted logic) of a vocabulary. For the example reasoning pattern (i.e., 'reasoning by assumption' in a game of Master Mind), the state ontology was inspired by the model depicted in Figure 1, and includes unary relations such as assumed and rejected_code on sort ASSUMPTION and binary relations such as prediction_for on RESULT x ASSUMPTION. The sort ASSUMPTION includes specific functions for domain statements such as code(COLOUR, COLOUR, COLOUR). The complete ontology for this current domain is given in Table 1.

Reasoning state. A (reasoning) state for ontology Ont is an assignment of truth-values {true, false} to the set of ground atoms At(Ont). The set of all possible states for ontology Ont is denoted by STATES(Ont). A part of the description of an example reasoning state S is:

assumed(code(red, white, blue))	: true
prediction_for(answer(black, empty, empty), code(red, white, blue))	: true
observation_result_for(answer(white), code(red, white, blue))	: true
rejected_code(code(red, white, blue))	: false

RS is the sort of all reasoning states of the agent. For simplicity in the formulation of properties WS is the set of all substates of elements of RS, thus WS is the set of all world states. The standard satisfaction relation |== between states and state properties is used: S |== p means that state property p holds in state S. For example, in the reasoning state S above it holds S |== assumed(code(red, white, blue)).

Reasoning trace. To describe dynamics, explicit reference is made to time in a formal manner. A fixed time frame T is assumed which is linearly ordered. Depending on the application, for example, it may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers). A trace

γ over an ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of reasoning states γ_t ($t \in T$) in STATES(Ont). The set of all traces over ontology Ont is denoted by Γ(Ont), i.e., $\Gamma(Ont) = STATES(Ont)^T$. The set Γ(Ont) is also denoted by Γ if no confusion is expected.

Unary relations:	
focus_assumed(F:FOCUS)	The agent currently assumes F to be part of the solution.
assumed(A:ASSUMPTION)	The agent currently assumes A to be the solution.
rejected_code(A:ASSUMPTION)	The agent has rejected the assumption A.
rejected_focus(F:FOCUS)	The agent has rejected the focus assumption F.
Binary relations:	
prediction_for(R:RESULT, A:ASSUMPTION)	The agent predicts that if A is true, then R should be observable.
code_extension_for(A:ASSUMPTION, F:FOCUS)	The agent believes that if F is part of the solution, then the whole solution should be A.
to_be_observed_for(answer, A:ASSUMPTION)	The agent starts observing what is the answer for A.
observation_result_for(R:RESULT, A:ASSUMPTION)	The agent observes that R is the answer for the guess A.
Sorts:	
FOCUS	at(C:COLOUR, P:POSITION)
ASSUMPTION	code(C1:COLOUR, C2:COLOUR, C3:COLOUR)
RESULT	answer(P1:PIN, P2:PIN, P3:PIN)
COLOUR	{black, blue, brown, green, orange, red, white, yellow)
POSITION	{1, 2, 3}
PIN	{black, white, _}

Table 1. Ontology for reasoning in the Master Mind domain

Expressing dynamic properties. States of a trace can be related to state properties via the formally defined satisfaction relation |== between states and formulae. Comparable to the approach in situation calculus, the sorted predicate logic temporal trace language TTL is built on atoms such as state(γ, t) |== p, referring to traces, time and state properties. This expression denotes that state property p is true in the state of trace γ at time point t. Here |== is a predicate symbol in the language (in infix notation), comparable to the Holds-predicate in situation calculus. Temporal formulae are built using the usual logical connectives and quantification (for example, over traces, time and state properties). The set TFOR(Ont) is the set of all temporal formulae that only make use of ontology Ont. We allow additional language elements as abbreviations of formulae of the temporal trace language. The fact that this language is formal allows for precise specification of dynamic properties. Moreover, editors can and actually have been developed to support specification of properties. Specified properties can be checked automatically against example traces to find out whether they hold.

4. The Experiment

Participants. Thirty persons with different social background participated in the experiment. The group consisted of 19 males and 11 females. Their mean age was 28.2 years, with a standard deviation of 10.0.

Method. The participants were asked to solve a simplified game of Master Mind. Before starting the experiment, they were given the following instructions:

The opponent picks a secret code consisting of three pegs, each peg being one of eight colors. Your goal is to guess the exact positions of the colors in the code in as few guesses as possible. After each guess, the opponent gives you a score of exact and partial matches. For each of the pegs in your guess that is the correct color in the correct position, the opponent will give you an 'exact' point (represented by a black pin). If you score 3 black pins on a guess, you have guessed the code. For each of the pegs in the guess that is a correct color in an incorrect position, the opponent will give you an 'other' point (represented by a white pin). Together, the black and white pins will add up to no more than 3. Notice that the positions of the black and white pins do not necessarily relate to the positions of the colors. Within this specific experiment, ***one initial guess has already been done for you.*** *While doing the experiment, please think aloud, explaining each step you perform.*

For each participant, the *solution code* was the same, namely the combination [blue-white-red]. The *initial guess* mentioned above was always the combination [red-white-blue]. Hence, the provided answer corresponding to the initial guess was [black-white-white].

In Table 2 and 3 two example traces are shown, and the way in which they were formalised in order to automatically check their properties. The left column contains the human transcript, the right column contains the formal counterpart. Two additional examples can be found in Appendix C. The transcripts of all human reasoning traces can be found at the following URL: http://www.cs.vu.nl/~tbosse/mastermind/human-traces.doc.

Human transcript	**Formalisation**
So, this is the first guess. Right. The national flag of Holland. *Exactly.*	
And this means that one of the colors is in the good place…ánd good color ánd good place, and also the other two colors are correct but they are not in the good place. *Exactly.*	
Right? Okay. So, what I'm going to do now. I'm going to…I'm trying to find out which of the colors is in a good place, first. So, let's say I say it's the red one. Maybe.	focus_assumed(at(red, 1))
So, I'm going to put the red here. And then, change these two.	code_extention_for(code(red, blue, white), at(red, 1)) assumed(code(red, blue, white)) prediction_for(answer(black, black, black), code(red, blue, white))
[red-blue-white] *Okay, so this is your guess?* This is my guess.	to_be_observed_for(answer, code(red, blue, white))
Then my answer is like this… *[white-white-white]* *…two, and three.*	observation_result_for(answer(white, white, white), code(red, blue, white))
Okay, so it wasn't the red. Okay.	rejected_code(code(red, blue, white)) rejected_focus(at(red, 1))
I will always use these ones, apparently. Then, keep the white and exchange red and blue.	focus_assumed(at(white, 2)) code_extention_for(code(blue, white, red), at(white, 2)) assumed(code(blue, white, red)) prediction_for(answer(black, black, black), code(blue, white, red))
[blue-white-red] *Okay, so why do you do this?* I'm testing now if the white one is in the good position.	to_be_observed_for(answer, code(blue, white, red))
Okay. So then my answer is this. *Congratulations!* *[black-black-black]*	observation_result_for(answer(black, black, black), code(blue, white, red))

Table 2. Example human reasoning trace

Human transcript	**Formalisation**
Well, at least the colours have already been determined.	
I want to know now…whether the red one was positioned correctly. *Okay.*	focus_assumed(at(red, 2))
[brown-red-__] Let's think now. Is this logical? I could of course also use one of the other colours twice. What would happen then? Then I can… Let's just see what happens then.	code_extension_for(code(blue, red, blue), at(red, 2)) assumed(code(blue, red, blue))
[blue-red-blue]	to_be_observed_for(answer, code(blue, red, blue))
These ones? Then the answer is as follows… *[black-white]*	observation_result_for(answer(black, white), code(blue, red, blue))
We know now that the white one was not placed correctly. No, we don't know that. Let's have a look, do we know that? No, we are not sure about that. It can also be that the blue one was in the right position, and that the white one was in the right position before that. Then it is the question whether this was a useful choice. At least it is the case that either…let's see now, if the red one was in the right position, then now the blue one is in the right position.	rejected_code(code(blue, red, blue))
But let me make the assumption that the white one was not in the right position.	focus_assumed(at(white, 1))
Then I would now…try this.	code_extension_for(code(white, red, blue), at(white, 1)) assumed(code(white, red, blue)) prediction_for(answer(black, black, black), code(white, red, blue))
[white-red-blue]	to_be_observed_for(answer, code(white, red, blue))
Then the answer is as follows… *[white-white-white]* *Like this.*	observation_result_for(answer(white, white, white), code(white, red, blue))
So nothing in the right position.	rejected_code(code(white, red, blue))
Let's see again. That means that in the first one the blue one was not in the right position either. So the blue one must be in the first or in the second.	rejected_focus(at(blue, 3))
The red one is not in the second...	rejected_focus(at(red, 2))
...so in the second only a blue one can have been right; that one is positioned in the first or in the second, so the blue one is on the first, that one is correct. So that we know already.	focus_assumed(at(blue, 1))
Furthermore, considering the white one. If the blue one should have been in the first, and we know that then…let's have a look, then there can be…then the white one has to be, according to that first one, to that first one it has to be correct.	code_extension_for(code(blue, white, red), at(blue, 1)) assumed(code(blue, white, red)) rejected_focus(at(white, 1))
Therefore this should be the solution.	prediction_for(answer(black, black, black), code(blue, white, red))
[blue-white-red]	to_be_observed_for(answer, code(blue, white, red))
All right. That is correct. *[black-black-black]* Good.	observation_result_for(answer(black, black, black), code(blue, white, red))

Table 3. Example human reasoning trace

5. Dynamic Properties

In this section a number of dynamic properties that have been identified as relevant for patterns of reasoning by assumption are presented. As mentioned in the Introduction, two categories of dynamic properties are distinguished. The first category is specified by *characterising properties*. These are properties that are expected to hold for all reasoning traces. In contrast, the second category contains *discriminating properties*, properties that distinguish several types of traces from each other. Within each category, *global properties* (GP's, addressing the overall reasoning behaviour) as well as *local properties* (LP's, addressing the step by step reasoning process) are given. Note that the properties are not given in any particular order, and that their numbering has no special meaning.

5.1. Characterising Properties

Based on the model presented in Section 2.4, a number of *characterising properties* have been expressed for the pattern of reasoning by assumption. These properties are shown below, both in an informal and a formal notation (in TTL).

GP1 Termination of Assumption Determination
The generation of new assumptions will not go indefinitely.
$\forall \gamma{:}\Gamma\ \exists t{:}T\ \forall A{:}INFO\_ELEMENT$
$\quad \forall t'{:}T \geq t{:}T \quad [\ state(\gamma,t') \mid\!\!== assumed(A) \Rightarrow$
$\quad\quad state(\gamma,t) \mid\!\!== assumed(A)\ ]$

This property holds for all traces, which is not surprising, since the experiments did not last forever.

GP2 Correctness of Rejection
Every code that has been rejected does not hold in the world situation.
$\forall \gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT$
$\quad state(\gamma,t) \mid\!\!== rejected\_code(A) \Rightarrow$
$\quad\quad state(\gamma,t) \mid\!\!=/= holds\_in\_world\_for(answer(black, black, black), A)$

This property holds for all traces, leading to the conclusion that none of the participants makes the error of rejecting a code that is actually the solution. However, this does not necessarily imply that none of the participants rejects partial information. To find out whether this is the case, an additional property should be needed, concentrating on rejected_focus instead of rejected_code.

GP3 Completeness of Rejection
After termination, all assumptions that do not hold in the world situation have been rejected.
$\forall \gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT$
$\quad termination(\gamma,t)$
$\quad \wedge\ state(\gamma,t) \mid\!\!== assumed(A)$
$\quad \wedge\ state(\gamma,t) \mid\!\!=/= holds\_in\_world\_for(answer(black, black, black), A)$
$\quad \Rightarrow state(\gamma,t) \mid\!\!== rejected\_code(A)$

Here $termination(\gamma, t)$ is defined as $\forall\ t'{:}T \quad t' \geq t \ \Rightarrow state(\gamma, t) = state(\gamma, t')$.

This property holds for all traces, implying that all participants eventually reject their incorrect assumptions. However, note that some of these rejections were made implicitly. For instance, consider the situation that a participant first assumes that the code is [red-blue-white], and subsequently assumes that the code is [blue-white-red]. In that case, the

predicate rejected_code(red, blue, white) was included in the trace, whilst the participant did not state this explicitly.

GP4 Guaranteed Outcome
After termination, at least one evaluated assumption has not been rejected.
$\forall\gamma{:}\Gamma\ \forall t{:}T$
  termination(γ,t) $\Rightarrow$ [$\exists$A:INFO_ELEMENT
    state(γ,t) |== assumed(A) $\wedge$ state(γ,t) |=/= rejected_code(A)]

This property holds for all traces, which indicates that every participant eventually finds the solution.

LP3 Observation Initiation Effectiveness
For each prediction an observation will be made.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT$
  state(γ,t) |== prediction_for(B,A)
  $\Rightarrow$ [$\exists t'{:}T \geq t{:}T$ state(γ,t') |== to_be_observed_for(answer, A)]

This property holds for all traces, leading to the conclusion that in every case that a prediction was made, this was followed by a corresponding observation.

LP4 Observation Result Effectiveness
If an observation is made the appropriate observation result will be received.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT$
  state(γ,t) |== to_be_observed_for(answer, A) $\wedge$
  state(γ,t) |== holds_in_world_for(B,A)
  $\Rightarrow$ [$\exists t'{:}T \geq t{:}T$ state(γ,t') |== observation_result_for(B,A)]

This property holds for all traces. Thus, in all traces, the opponent provided the correct answers.

LP5 Evaluation Effectiveness
If an assumption was made and a related prediction is falsified by an observation result, then the assumption is rejected.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT$
  state(γ,t) |== assumed(A) $\wedge$ state(γ,t) |== prediction_for(B,A)
  $\wedge$ state(γ,t) |== observation_result_for(C,A) $\wedge$ $B \neq C$
  $\Rightarrow$ [$\exists t'{:}T \geq t{:}T$ state(γ,t') |== rejected_code(A)]

This property, which relates to GP2, holds for all traces. Thus, all participants correctly rejected a certain assumption when they had reason to do this (i.e., when the corresponding prediction was falsified by an observation result).

5.2. Discriminating Properties

An analysis in terms of characterising properties as mentioned above is useful to create and validate a generic theory on a specific type of reasoning. Here, by generic it is meant that the theory can be applied to any particular person who reasons by assumption, regardless of the specific strategy used. However, usually in reasoning tasks also differences can be observed between individuals. Therefore, it is useful to also specify a number of *discriminating properties* of reasoning by assumption. These properties are shown below, both in an informal and a formal notation. In addition, for each property it is mentioned for how many of the 30 participants the property turned out to hold.

GP5 Correctness of Assumption
Everything that has been assumed holds in the world situation.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT$
  state(γ,t) |== assumed(A) $\Rightarrow$
    state(γ,t) |== holds_in_world_for(answer(black, black, black), A)

This property only holds in four of the 30 cases. By checking it, the participants that made only correct assumptions can be distinguished from those that made some incorrect assumptions during the experiment. Put differently, the participants that immediately make the right guess are distinguished from those that need more than one guess. The fact that only four of the 30 participants are successful in their first guess indicates (as could be expected) that there is no confounding in the experiment whereby the participants can pick up information about the correct guess.

GP6 Assumption Grounding

Everything that has been assumed was based on an underlying focus (and code extension).

$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT$
  $state(\gamma,t) \mid\!\!= assumed(A)$
  $\Rightarrow [\ \exists t'{:}T < t{:}T\ \ \exists B{:}INFO\_ELEMENT\ \ state(\gamma,t') \mid\!\!=$
    $focus\_assumed(B) \wedge state(\gamma,t') \mid\!\!= code\_extension\_for(A,B)\ ]$

This property holds in 26 of the 30 cases. Hence, the majority of the participants always generate their assumptions in two steps: first, they assume a certain color for one of the three positions, and then they extend this focus with assumptions for the other two positions. In contrast, four cases were found where the participants did not reason this way. These participants assumed a certain code without an underlying focus. There are two possible explanations for this phenomenon. One is that they did in fact make the focus assumption internally, but that this could not be derived with certainty from their externally observable behavior. The second explanation is that they did not quite understand the rules of the game, and hoped to make some progress by simply choosing a random code.

GP7 Observation Effectiveness

For each assumption, the agent eventually obtains the appropriate observation result.

$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT$
  $state(\gamma,t) \mid\!\!= assumed(A) \wedge state(\gamma,t) \mid\!\!= holds\_in\_world\_for(B,A)$
  $\Rightarrow\ [\ \exists t'{:}T \geq t{:}T\ \ \ state(\gamma,t') \mid\!\!= observation\_result\_for(B,A)\ ]$

This property states that the agent always obtains the appropriate observation result for a particular assumption. For example, if an assumption is completely correct in the world, then the appropriate observation result should be three times black. Thus, the property gives more information about the experimenter than about the participant. In the experiments, this property holds for all but three of the traces. In these three cases people make an assumption that cannot be right, according to the information they have. However, they correct themselves before they decide to observe the answer to this wrong assumption. Thus, the answer to the incorrect assumption is never obtained.

GP8 Essential Assumption

When a solution has been found, this was due to the focus at(white, 2).

$\forall\gamma{:}\Gamma\ \forall t{:}T$
  $termination(\gamma,t) \wedge state(\gamma,t) \mid\!\!= assumed(code(blue, white, red))$
  $\Rightarrow [\ \exists t'{:}T < t{:}T$
    $state(\gamma,t') \mid\!\!= focus\_assumed(at(white, 2)) \wedge$
    $state(\gamma,t') \mid\!\!= code\_extension\_for(code(blue, white, red),at(white, 2))\ ]$

This property holds in 25 of the 30 cases. Thus, the majority of the participants found the solution, [blue-white-red], thanks to the assumption that the white pin was at position 2. However, other strategies are used as well, e.g. focussing on the red or the blue pin.

GP9 Initial Assumption
The first focus assumption made was at(red, 1).

$\forall\gamma{:}\Gamma\ \exists t{:}T$
  state(γ,t) |== focus_assumed(at(red, 1))
  $\wedge$ [$\forall t'{:}T < t{:}T$ $\forall A{:}$INFO_ELEMENT
    state(γ,t') |== focus_assumed(A) $\Rightarrow$ A = at(red, 1)]

This property holds in 18 of the 30 cases. Thus, 18 participants started reasoning by assuming that the red pin was at position 1. There are two possible explanations for this overall preference. First, although it is stated in the experiment that the order of the evaluation pins has no meaning, some of the participants might be guided by this order (i.e., black-white-white) in the first guess. Second, some participants might have a preference to analyse the pins systematically from left to right, and therefore start by focussing on the red pin. Nevertheless, there were still 12 participants that started in a different way.

GP10 Second Assumption
The second focus assumption made was at(red, 2).

$\forall\gamma{:}\Gamma\ \exists t{:}T$
  second_focus(γ,t) $\wedge$
  state(γ,t) |== focus_assumed(at(red, 2))

Here second_focus(γ, t) is defined as
$\exists A{:}$INFO_ELEMENT state(γ,t) |== focus_assumed(A) $\wedge$
$\exists t'{:}T < t{:}T$ $\exists B{:}$INFO_ELEMENT state(γ,t') |== focus_assumed(B) $\wedge$
[$\forall t''{:}T < t{:}T$ $\forall C{:}$INFO_ELEMENT state(γ,t'') |== focus_assumed(C) $\Rightarrow$ C = B]

This property holds in 3 of the 30 cases. This means that three participants based their second guess upon the focus assumption at(red, 2). In fact, all of these three participants based their first guess upon the focus assumption at(red, 1). Thus, in the first two guesses they consistently focussed on the position of the red pin. This is an important property, since it distinguishes two different types of reasoners with respect to the second guess: those that keep their focus on red (but realise that it has to be in another position) versus those that shift to another colour. Although both approaches eventually lead to the same solution, the difference is relevant, since the reasoning strategies used are clearly distinct.

LP2 Prediction Effectiveness
For each assumption that is made a prediction will be made.

$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}$INFO_ELEMENT
  state(γ,t) |== assumed(A)
  $\Rightarrow$ [$\exists t'{:}T \geq t{:}T$ $\exists B{:}$INFO_ELEMENT state(γ,t') |== prediction_for(B,A)]

This property holds in 26 of the 30 cases. So in four cases the participants make an assumption for which no prediction is made. Three of these four traces have already been discussed at GP7. The fourth trace involves the situation of Table 3, where the participant uses the following reasoning pattern: "… I could use one of the colors twice. What would happen then? Well, I don't know. Let's just see what happens…" Hence, the participant tries a code of which he intuitively thinks that it is an intelligent guess, without really understanding why. Therefore, he does not make a prediction.

LP2' Prediction Optimism
For each assumption that is made the prediction answer(black, black, black) will be made.

$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}$INFO_ELEMENT
  state(γ,t) |== assumed(A)
  $\Rightarrow$ [$\exists t'{:}T \geq t{:}T$
      state(γ,t') |== prediction_for(answer(black, black, black),A)]

This property is a variant of property LP2. It holds in 24 of the 30 cases. In these cases the participants predict for every assumption they make, that it is the correct solution. Given the fact that the participants have no special talent for guessing (see GP5), it might be a bit surprising that so many of them still make guesses of which they 'hope' they are correct, rather than using a more systematic strategy. See Section 8 for a more detailed discussion upon this topic. Nevertheless, for 6 participants property LP2' does not hold. Four of these six participants are those that make no predictions at all (see LP2). The interesting cases, however, are the two participants that do make predictions, but that predict that their assumptions are *not* entirely correct. It turned out that this way of reasoning was part of a deliberate strategy of the participants. What they did was making a focus assumption (e.g. a red pin is at position 1), and then extending this focus by adding 'neutral' colors (e.g. assuming the code [red-yellow-yellow]). By doing this, the participant already knows that her guess will not be entirely correct, but she still makes this guess in order to receive partial information of the solution in a very systematic way.

6. Results

A special piece of software has been developed that takes a formally specified property and a set of traces as input, and verifies whether the property holds for the traces (see Bosse *et al.*, 2004). By means of this checking software, all specified properties have been checked automatically against all traces to find out whether they hold. In Table 4 and 5 an overview of the results is shown. In these tables, an X indicates that the property holds for that particular trace. The final row provides the number of guesses needed by each participant to solve the problem.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GP1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GP2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GP3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GP4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GP5	X	-	-	-	-	-	-	-	-	-	-	-	-	X	-
GP6	X	X	X	X	-	-	X	X	X	X	X	X	X	X	X
GP7	X	X	X	X	X	-	X	X	X	X	X	X	X	X	-
GP8	X	X	-	X	X	X	X	X	X	-	X	X	X	X	X
GP9	-	-	-	-	-	X	X	X	X	-	X	X	X	-	-
GP10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LP3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LP4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LP5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LP2	X	X	X	X	X	-	X	X	X	-	X	X	X	X	-
LP2'	X	X	-	X	X	-	X	X	X	-	X	X	X	X	-
steps	1	2	3	3	3	3	3	2	3	3	2	3	2	1	3

Table 4. Overview of the experimental results (1): traces against properties

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
GP1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GP2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GP3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GP4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
GP5	-	-	-	X	-	-	-	-	-	-	X	-	-	-	-
GP6	X	X	-	X	X	X	X	X	X	X	X	X	-	X	X
GP7	X	X	-	X	X	X	X	X	X	X	X	X	X	X	X
GP8	X	X	-	X	X	X	X	-	X	X	X	X	-	X	X
GP9	X	X	X	-	X	X	X	X	-	X	-	-	X	X	X
GP10	X	-	-	-	-	-	-	X	-	-	-	-	X	-	-
LP3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LP4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LP5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LP2	X	X	-	X	X	X	X	X	X	X	X	X	X	X	X
LP2'	X	X	-	X	X	-	X	X	X	X	X	X	X	X	X
steps	3	3	3	1	3	2	2	3	3	2	1	3	3	3	2

Table 5. Overview of the experimental results (2): traces against properties

As can be seen in these tables, all characterising properties indeed hold for all traces. This contributes to the validation of the model presented in Section 2.4. However, note that this is only an empirical validation, based on a limited number of empirical traces.

As opposed to the characterising properties, the discriminating properties only hold for some of the traces. Therefore, these results can be used to distinguish several types of transcripts from each other, thereby obtaining a classification of different reasoning strategies. To do this in a more structured way, some simple Tree Clustering techniques (Kaufman and Rousseeuw, 1990) have been used to reduce the number of different classes. In order to do this, the following procedure was used. In the first step, all rows indicating characterising properties have been removed from the tables, and the resulting individuals with the same properties have been clustered together. In the following steps, more rows have been removed from the tables (in a stepwise manner, starting with the discriminating property that holds for most individuals, i.e., GP7). This process has been repeated until only four rows were left. The results can be seen in Table 6. These results suggest that most of the reasoners fall in the fourth class (for which the properties GP8, GP9 and LP2' hold, and GP10 does not hold). This class of reasoners could be described as the 'systematic reasoners', since their reasoning processes satisfy the following combination of properties:

- They start focussing on the left pin (GP9)
- They continue by focussing on another colour that could have corresponded with the black pin (instead of focussing on red again, GP10)
- They only make guesses that are possible solutions (LP2')
- They find a solution due to the focus on the white pin (GP8)

Another interesting class of reasoners is defined by all traces for which property LP2' does not hold (i.e., a combination of column 2, 3, 5, and 7). This class of reasoners follows a rather specific strategy. They all start by using 'wrong' colours in order to obtain information about part of the solution. Only after obtaining this partial information,

they start making guesses about the solution as a whole. Therefore, this class could be described as the 'stepwise' reasoners. In a similar manner, some qualifications could be given to the other classes, such as 'strategic reasoners' or 'random reasoners'.

	1,2,4,5,14,19,24,26,27	3,10	6,21	7,8,9,11,12,13,17,20,22,25,29,30	15	16	18	23,28
GP8	X	-	X	X	X	X	-	-
GP9	-	-	X	X	-	X	X	X
GP10	-	-	-	-	-	X	-	X
LP2'	X	-	-	X	-	X	-	X

Table 6. Overview of the results after applying Tree Clustering

7. Logical Relationships

In addition to the above, logical relationships have been identified between properties at different abstraction levels. An overview of the identified logical relationships relevant for overall property GP7 is depicted as an AND-tree in Figure 2.

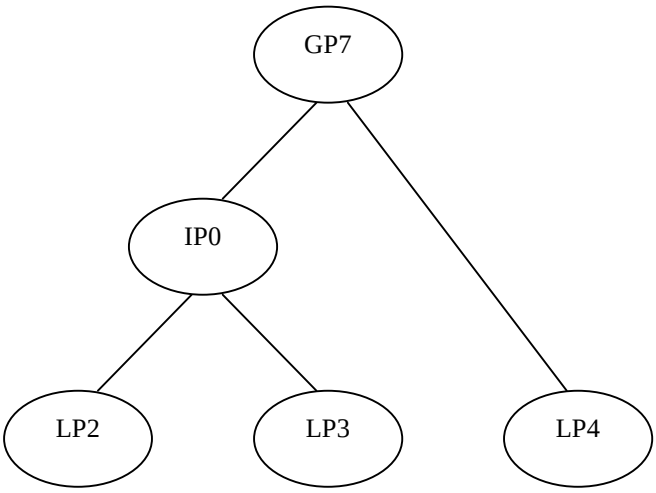


Figure 2. Logical relationships between dynamic properties

For example, the relationship at the highest level expresses that IP0& LP4 => GP7 holds. Here, IP0 is an *intermediate property*, expressing the dynamics of the reasoning between two milestones:

IP0 Assumptions lead to Observation Initiation
For each assumption that is made a prediction will be made.
$\forall \gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT$
state(γ,t) |== assumed(A)
$\Rightarrow$ [$\exists t'{:}T \geq t{:}T$ state(γ,t') |== to_be_observed_for(answer, A)]

Intermediate properties address smaller steps than global properties do, but bigger steps than local properties do. At a lower level, Figure 2 depicts the relationship LP2 & LP3 => IP0.

Notice that the results given in Table 4 and 5 validate these logical relationships. For instance, in all traces where LP2, LP3 and LP4 hold, also GP7 holds. Such logical

relationships between properties can be very useful in the analysis of empirical reasoning processes. For example, if a given person does not obtain the appropriate observation result for her assumption (i.e. property GP7 is not satisfied by the reasoning trace), then by a refutation process it can be concluded that either property IP0, or property LP4 fails (or both). If, after checking these properties, it turns out that IP0 does not hold, then either LP2 or LP3 does not hold. Thus, by this example refutation analysis it can be concluded that the cause of the unsatisfactory reasoning process can be found in either LP2 or LP3. In other words, either the *Observation Initiation* mechanism fails (LP3), or the *Prediction* mechanism fails (LP2).

In this section, only one logical relationship is shown. However, many more global, intermediate, and local properties for the pattern of reasoning by assumption, as well as the relationships between them can be found in Appendix A and B.

8. Discussion

Within our experiment, the number of guesses needed by the participants in order to solve the problem varied between one and three. However, the participants that only needed one guess did not know beforehand that their guess would be correct. They were just lucky, since other solutions were possible, given the initial situation. Thus, their strategy was not optimal. Nevertheless, a number of studies exist that analyse optimal strategies in Master Mind; for example, (Knuth, 1977; Koyama and Lai, 1994). With respect to our specific (simplified) problem, it turns out that there are optimal strategies that can always solve the problem in two guesses. In order to apply such a strategy, one should start with a code involving one of the initial colors twice. For instance, [red-red-blue]. Making this guess will provide enough information to solve the problem in the next guess. The reason for this is that, given the initial situation, only three solutions are possible, namely [red-blue-white], [blue-white-red] and [white-red-blue]. And for each of these possible solutions, the guess [red-red-blue] will receive a unique answer, i.e. [black-white], [white-white], and [black-black], respectively.

Given the above, a natural question is why none of the participants used this optimal strategy. A first possible reason is that it seems unnatural for humans to make a guess of which they know beforehand that it will not be the correct solution. Starting in the way as described above would feel like wasting a guess. This might explain part of the results, but as discussed in Section 6, some of the participants (the ‘stepwise’ reasoners) did deliberately use some ‘wrong’ colours (although they failed to discover the optimal strategy). So the question remains why these stepwise reasoners did not find the optimal strategy. A potential answer is that it appears to be difficult (or at least, not very attractive from a work load perspective) for the participants to start by exhaustively generating all possible solutions. If they would do that, they would find out that the problem in question is probably simpler than they expected, involving only three possible solutions. However, the work load needed to find this out is relatively high compared to the gain (of just one step). Still, a small subset of the participants did generate all possible solutions, but even they did not come up with an optimal strategy. Therefore some other inhibitory factors may have played a role as well. Examples of such factors are time and social pressure (some participants might be embarrassed when spending too much time on a rather simple problem) and motivational factors (the participants were not informed that the optimal solution could be found in two steps, so they were not really encouraged to try harder).

A final factor that may have played a role in the strategy selection of the participants, is the specific domain of Master Mind. Possibly, in other application domains of the pattern of ‘reasoning by assumption’ other strategies are preferred. For example, in the domain of

diagnosis the strategy of only making guesses that are expected to hold (see property LP2') could be less attractive. In this domain, people may be more likely to make assumptions that are expected *not* to hold, thereby eliminating causes in a systematic way (rule out strategy). More research is needed to determine the extent to which the results found in this paper can be generalised to other applications of 'reasoning by assumption'.

9. Conclusion

This paper introduces a novel approach for the analysis of reasoning processes, and explores the applicability of the approach for the pattern of 'reasoning by assumption' in the domain of Master Mind. The analysis approach is based on the formalisation of empirical reasoning traces, and the automated analysis of dynamic properties. A variety of dynamic properties have been specified, some of which are considered characteristic for the reasoning pattern 'reasoning by assumption', whereas some other properties can be used to discriminate between different approaches to the reasoning. For the Master Mind experiments undertaken, properties of the first, characteristic, type were based on the basis of the model from (Jonker and Treur, 2003). These properties indeed turned out to hold for the acquired reasoning traces, which contributes to the empirical validation of the model. Properties of the latter, discriminating type hold for some of the traces and do not hold for other traces: they define subsets of traces that collect similar reasoning approaches. These subsets can be viewed as different classes of reasoners, such as systematic reasoners, stepwise reasoners, strategic reasoners and random reasoners. In the current experiments, the biggest class of reasoners was the 'systematic' class. These persons started by focussing on the left pin, continued by focussing on the white or blue pin, and eventually found the solution due to a focus on the white pin. Moreover, during the whole experiment they only made guesses that are possible solutions. Nevertheless, several other strategies were observed. An interesting class was the class of 'stepwise' reasoners, which tried to obtain partial information in a stepwise manner. Future research is necessary to find out whether these results are specific for the game of Master Mind, or whether they can be generalised to other applications of 'reasoning by assumption'.

In addition to the above, it was explained how logical relationships can be established between dynamic properties at different levels (e.g. global dynamic properties are connected to local dynamic properties, via intermediate properties). It was shown that such interlevel relationships may play an important role in the analysis of empirical reasoning processes. More specifically, it was shown how a refutation process can be used to localise the exact cause of failure of global properties that are expected to hold.

In addition to empirical traces, the analysis approach presented in this paper can be applied to traces generated by simulation models. Dynamic properties found relevant for human traces can be used to validate a simulation model, by generating a number of simulation runs and checking the dynamic properties for the resulting traces. This type of validation has been exploited to validate a simulation model for reasoning by assumption to solve the wise men puzzle in (Jonker and Treur, 2003). Moreover, in (Bosse, Jonker and Treur, 2003) a similar analysis approach has been used to validate a simulation model for controlled multi-representational reasoning involving arithmetic, geometric and material representations.

Besides Cognitive Science, the analysis method can be relevant for the area of Knowledge Engineering. The aim in Knowledge Engineering is to (formally) model complex reasoning tasks, such as design or diagnosis. This contributes to modelling, design, evaluation, maintenance, validation and verification, and reuse of models (Fensel and van Harmelen, 1994; Treur and Wetter, 1993). Some previous work in Knowledge

Engineering in the domain of problem solving is reported by (Brazier *et al.*, 1999). In their paper, the relevant domain knowledge is obtained mainly by means of interviews with domain experts. The present work can be viewed as complementary to their work, because here the relevant domain knowledge is obtained by means of explicit experiments with a large number of participants.

With respect to future research, an interesting direction would be to observe the participants' reasoning behaviour over multiple trials. Important questions in this respect are whether participants are able to discover and learn certain strategies, and whether experienced puzzlers perform better than novices. To answer these kinds of questions, for future work it is planned to perform a learning experiment where participants have to solve multiple puzzles at different time points.

Another possibility for further research is to compare the current work with the work by (Stenning and van Lambalgen, 2005). In that paper, the authors show that the two traditional approaches for modelling reasoning (the syntactic and the semantic approach) are not as mutually exclusive as they are often presented. In line with their claims, the current paper does not make any commitments to one of both approaches either. Instead, it introduces a generic approach to analyse the dynamics of reasoning processes, no matter whether these are represented by 'rules' or by 'models'. Moreover, Stenning and van Lambalgen continue by proposing an alternative distinction in reasoning processes, i.e., a distinction between reasoning *towards* an interpretation and reasoning *from* an interpretation to a conclusion. They demonstrate that this distinction is more appropriate to explain empirical findings in reasoning, such as the suppression effect (Byrne, 1989). It remains to be investigated how this distinction connects with the current research. One difference between our Master Mind experiment and the type of tasks considered in (Stenning and van Lambalgen, 2005) is that in the latter the relevant external information is given in natural language (i.e., a number of sentences), whereas in the former it has a more 'mathematical' format (i.e., six coloured pins). Therefore, in the type of reasoning modelled in this paper the process of interpretation is less present (there is less room for different interpretations), so that it involves mainly reasoning *from* a (fixed) interpretation. Nevertheless, even in the Master Mind example there is still some reasoning *to* an interpretation. To investigate in more detail what is the role of interpretation in reasoning by assumption, it would be interesting to change the setup of the experiments in such a way that some more explicit interpretation is needed.

Acknowledgements

The authors are grateful to James Greeno, Keith Stenning and an anonymous referee for their valuable comments on an earlier version of this paper.

Appendix A. Dynamic Properties

This Appendix contains a number of dynamic properties that are relevant for the pattern of reasoning by assumption. All of the global properties and a random selection of the intermediate properties have been validated against the traces mentioned in Section 6, using automated checks as described in that section. Note that in some cases the terminology used in these properties does not completely match the terminology used in the properties given in Section 5. The reason for this is that the properties given in Section 5 are domain-specific: they apply to the domain of Master Mind only, whereas the properties given here apply to the pattern of reasoning by assumption in general. For

example, a number of them have been checked against (human and simulation) traces in another case study involving reasoning by assumption: the wise men puzzle (Jonker and Treur, 2003).

World assumptions

WP1 World consistency
If something holds in the world, then its complement does not hold.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
state(γ,t) |== holds_in_world(A,S1) $\wedge$ S1 $\neq$ S2 $\Rightarrow$ state(γ,t) |=/= holds_in_world(A,S2)
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
state(γ,t) |=/= holds_in_world(A,S1) $\Rightarrow$ state(γ,t) |== holds_in_world(A,S2) $\wedge$ S1 $\neq$ S2

Domain assumptions

DK1 Domain knowledge correctness
All domain knowledge about assumptions implying predictions is correct.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
state(γ,t) |== holds_in_world(A,S1) $\wedge$ domain_implies(A,S1,B,S2)
$\Rightarrow$ state(γ,t) |== holds_in_world(B,S2)]
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
state(γ,t) |=/= holds_in_world(A,S1) $\wedge$ domain_implies(A,S1,B,S2)
$\Rightarrow$ state(γ,t) |=/= holds_in_world(B,S2)]

Local properties

LP1 Assumption initialisation
Make a first assumption.
$\forall\gamma{:}\Gamma\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
initial_assumption(A,S)
$\Rightarrow$ [$\exists$ t:T state(γ,t) |== assumed(A,S)]

LP2 Prediction effectiveness
For each assumption that is made all relevant predictions are generated.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
state(γ,t) |== assumed(A,S1) $\wedge$ domain_implies(A,S1,B,S2)
$\Rightarrow$ [$\exists$t':T $\geq$ t:T state(γ,t') |== prediction_for(B,S2,A,S1)]

LP3 Observation initiation effectiveness
All predictions made will be observed.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
state(γ,t) |== prediction_for(B,S2,A,S1)
$\Rightarrow$ [$\exists$t':T $\geq$ t:T state(γ,t') |== to_be_observed(B)]

LP4 Observation result effectiveness
If an observation is made the appropriate observation result will be received.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
state(γ,t) |== to_be_observed(A) $\wedge$ state(γ,t) |== holds_in_world(A,S)
$\Rightarrow$ [$\exists$t':T $\geq$ t:T state(γ,t') |== observation_result(A,S)]

LP5 Evaluation effectiveness
If an assumption was made and a related prediction is falsified by an observation result, then the assumption is rejected.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2,S3{:}SIGN$
state(γ,t) |== assumed(A,S1) $\wedge$ state(γ,t) |== prediction_for(B,S2,A,S1)
$\wedge$ state(γ,t) |== observation_result(B,S3) $\wedge$ S2 $\neq$ S3
$\Rightarrow$ [$\exists$t':T $\geq$ t:T state(γ,t') |== rejected(A,S1)]

LP6 Assumption effectiveness
If an assumption is rejected, and there is still an alternative assumption available, this will be assumed.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \mid\!\!= assumed(A,S1)$
$\wedge\ state(\gamma,t) \mid\!\!= rejected(A,S1)$
$\wedge\ state(\gamma,t) \mid\!\!= alternative\_for(B,S2,A,S1)$
$\wedge\ state(\gamma,t) \mid\!\!\neq rejected(B,S2)$
$\Rightarrow\ [\ \exists t'{:}T \geq t{:}T\quad state(\gamma,t') \mid\!\!\neq assumed(A,S1) \wedge state(\gamma,t') \mid\!\!= assumed(B,S2)\ ]$

Global properties

GP1 Termination of assumption determination
The generation of new assumptions will not go indefinitely.
$\forall\gamma{:}\Gamma\ \exists t{:}T\ \forall A{:}INFO\_ELEMENT,\ \forall S{:}SIGN$
$\forall t'{:}T \geq t{:}T\quad [\ state(\gamma,t') \mid\!\!= assumed(A,S) \Rightarrow state(\gamma,t) \mid\!\!= assumed(A,S)\ ]$

GP2 Correctness of rejection
Everything that has been rejected does not hold in the world situation.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \mid\!\!= rejected(A,S)$
$\Rightarrow state(\gamma,t) \mid\!\!\neq holds\_in\_world(A,S)$

GP3 Completeness of rejection
After termination, all assumptions that do not hold in the world situation have been rejected.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT,\ \forall S{:}SIGN$
$termination(\gamma,t)$
$\wedge\ state(\gamma,t) \mid\!\!= assumed(A,S)$
$\wedge\ state(\gamma,t) \mid\!\!\neq holds\_in\_world(A,S)$
$\Rightarrow state(\gamma,t) \mid\!\!= rejected(A,S)$

P Persistence
Atoms are persistent (either unconditional or conditional).
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \mid\!\!= holds\_in\_world(A,S)$
$\Rightarrow [\ \forall t'{:}T \geq t{:}T\quad state(\gamma,t') \mid\!\!= holds\_in\_world(A,S)\ ]$
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \mid\!\!\neq holds\_in\_world(A,S)$
$\Rightarrow [\ \forall t'{:}T \geq t{:}T\quad state(\gamma,t') \mid\!\!\neq holds\_in\_world(A,S)\ ]$
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \mid\!\!= rejected(A,S)$
$\Rightarrow [\ \forall t'{:}T \geq t{:}T\quad state(\gamma,t') \mid\!\!= rejected(A,S)\ ]$
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \mid\!\!= observation\_result(A,S)$
$\Rightarrow [\ \forall t'{:}T \geq t{:}T\quad state(\gamma,t') \mid\!\!= observation\_result(A,S)\ ]$
$\forall\gamma{:}\Gamma\ \forall t,t',t''{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$t \leq t'' \wedge state(\gamma,t) \mid\!\!= assumed(A,S)$
$\wedge\ [\ t \leq t' \leq t'' \Rightarrow state(\gamma,t') \mid\!\!\neq rejected(A,S)\ ]$
$\Rightarrow state(\gamma,t'') \mid\!\!= assumed(A,S)$
$\forall\gamma{:}\Gamma\ \forall t,t',t''{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$t \leq t'' \wedge state(\gamma,t) \mid\!\!= prediction\_for(A,S1,B,S2)$
$\wedge\ [\ t \leq t' \leq t'' \Rightarrow state(\gamma,t') \mid\!\!\neq rejected(B,S2)\ ]$
$\Rightarrow state(\gamma,t'') \mid\!\!= prediction\_for(A,S1,B,S2)$

Intermediate properties

IP1 Assumption existence uniqueness (1)
An assumption is never assumed twice.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall t'{:}T{>}t{:}T\ \forall A{:}INFO\_ELEMENT, \forall S{:}SIGN$
$state(\gamma,t) \models\models assumed(A,S) \wedge state(\gamma,t') \models/\models assumed(A,S)$
$\Rightarrow [\ \forall t''{:}T > t'{:}T \quad state(\gamma,t'') \models/\models assumed(A,S)\ ]$

IP2 Possible assumption finiteness
There is a finite number N of possible assumptions.
$card(pa, N) \quad \equiv \quad \exists A_1 \ldots A_N\ [\ \wedge\wedge_{i\neq j}\ A_i \neq A_j \ \wedge\ pa(A_i) \ \wedge\ \forall A\ [\ pa(A) \Rightarrow \vee\vee_k\ A = A_k\ ]\ ]$

IP3 Assumption grounding
Each assumption that is assumed is a possible assumption.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \models\models assumed(A,S) \Rightarrow pa(A,S)$

IP4 Assumption retraction implies rejection
If something is assumed first, and later not assumed anymore, then it has been rejected.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall t'{:}T{>}t{:}T\ \forall A{:}INFO\_ELEMENT, \forall S{:}SIGN$
$state(\gamma,t) \models\models assumed(A,S) \wedge state(\gamma,t') \models/\models assumed(A,S)$
$\Rightarrow state(\gamma,t') \models\models rejected(A,S)$

IP5 Assumption existence uniqueness (2)
If something is rejected, then it will never be assumed again.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT, \forall S{:}SIGN$
$state(\gamma,t) \models\models rejected(A,S)$
$\Rightarrow [\ \forall t'{:}T > t{:}T \quad state(\gamma,t') \models/\models assumed(A,S)\ ]$

IP6 Proper rejection grounding
If an assumption is rejected, then earlier on there was a prediction for it that did not match the corresponding observation result.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S1{:}SIGN$
$state(\gamma,t) \models\models rejected(A,S1)$
$\Rightarrow [\exists t'{:}T \leq t{:}T\ \exists B{:}INFO\_ELEMENT\ \exists S2,S3{:}SIGN$
$state(\gamma,t') \models\models prediction\_for(B,S2,A,S1) \ \wedge state(\gamma,t') \models\models observation\_result(B,S3) \wedge S2 \neq S3]$

IP7 Prediction-observation discrepancy implies assumption incorrectness
If a prediction does not match the corresponding observation result, then the associated assumption does not hold in the world.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2,S3{:}SIGN$
$state(\gamma,t) \models\models prediction\_for(B,S2,A,S1) \wedge state(\gamma,t) \models\models observation\_result(B,S3) \wedge S2 \neq S3$
$\Rightarrow state(\gamma,t) \models/\models holds\_in\_world(A,S1)$

IP8 Observation result correctness
Observation results obtained from the world indeed hold in the world.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \models\models observation\_result(A,S) \ \Rightarrow \ state(\gamma,t) \models\models holds\_in\_world(A,S)$

IP9 An incorrect prediction implies an incorrect assumption (1)
If a prediction does not match the facts from the world, then the associated assumption does not hold either.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2,S3{:}SIGN$
$state(\gamma,t) \models\models prediction\_for(B,S2,A,S1) \wedge state(\gamma,t) \models\models holds\_in\_world(B,S3) \wedge S2 \neq S3$
$\Rightarrow state(\gamma,t) \models/\models holds\_in\_world(A,S1)$

IP10 Observation result grounding
If an observation has been obtained, then earlier on the corresponding fact held in the world.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \models\models observation\_result(A,S) \ \Rightarrow \ [\ \exists t'{:}T \leq t{:}T \quad state(\gamma,t') \models\models holds\_in\_world(A,S)\ ]$

IP11 An incorrect prediction implies an incorrect assumption (2)
If a prediction does not hold in the world, then the associated assumption does not hold either.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \models prediction\_for(B,S2,A,S1) \wedge state(\gamma,t) \not\models holds\_in\_world(B,S2)$
$\Rightarrow state(\gamma,t) \not\models holds\_in\_world(A,S1)$

IP12 Prediction correctness
If a prediction is made for an assumption that holds in the world, then the prediction also holds.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \models prediction\_for(B,S2,A,S1) \wedge state(\gamma,t) \models holds\_in\_world(A,S1)$
$\Rightarrow state(\gamma,t) \models holds\_in\_world(B,S2)$

IP13 Rejection effectiveness
If an assumption has been made and it does not hold in the world state, then it will be rejected.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT,\ \forall S{:}SIGN$
$state(\gamma,t) \models assumed(A,S)$
$\wedge\ state(\gamma,t) \not\models holds\_in\_world(A,S)$
$\Rightarrow [\exists t'{:}T \geq t{:}T \quad state(\gamma,t') \models rejected(A,S)\ ]$

IP14 An incorrect assumption implies prediction-observation discrepancy
If an assumption is made and it does not hold in the world, then a prediction for that assumption will be made that does not match the corresponding observation result.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT,\ \forall S1{:}SIGN$
$state(\gamma,t) \models assumed(A,S1)$
$\wedge\ state(\gamma,t) \not\models holds\_in\_world(A,S1)$
$\Rightarrow [\exists t'{:}T \geq t{:}T\ \exists B{:}INFO\_ELEMENT,\ \exists S2,S3{:}SIGN$
$state(\gamma,t') \models prediction\_for(B,S2,A,S1) \wedge state(\gamma,t') \models observation\_result(B,S3) \wedge S2 \neq S3\ ]$

IP15 An incorrect assumption implies an incorrect prediction (1)
If an assumption is made and it does not hold in the world, then a prediction for that assumption will be made that does not match the corresponding facts from the world.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT,\ \forall S1{:}SIGN$
$state(\gamma,t) \models assumed(A,S1)$
$\wedge\ state(\gamma,t) \not\models holds\_in\_world(A,S1)$
$\Rightarrow [\exists t'{:}T \geq t{:}T\ \exists B{:}INFO\_ELEMENT,\ \exists S2,S3{:}SIGN$
$state(\gamma,t') \models prediction\_for(B,S2,A,S1) \wedge state(\gamma,t') \models holds\_in\_world(B,S3) \wedge S2 \neq S3\ ]$

IP16 Observation effectiveness
For each prediction, the agent makes the appropriate observation.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT,\ \forall S1,S2,S3{:}SIGN$
$state(\gamma,t) \models prediction\_for(B,S2,A,S1) \wedge state(\gamma,t) \models holds\_in\_world(B,S3)$
$\Rightarrow [\exists t'{:}T \geq t{:}T \quad state(\gamma,t') \models observation\_result(B,S3)\ ]$

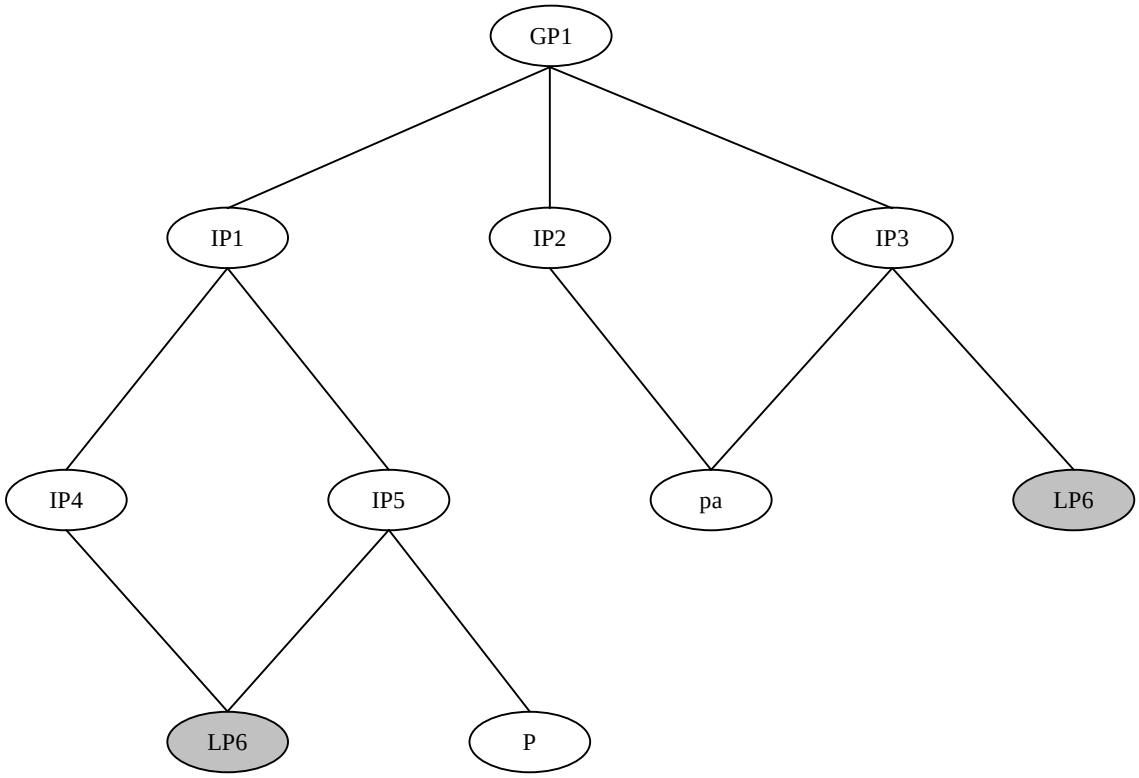
IP17 An incorrect assumption implies an incorrect prediction (2)
If an assumption is made and it does not hold in the world, then a prediction for that assumption will be made that does not hold either.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT,\ \forall S1{:}SIGN$
$state(\gamma,t) \models assumed(A,S1)$
$\wedge\ state(\gamma,t) \not\models holds\_in\_world(A,S1)$
$\Rightarrow [\exists t'{:}T \geq t{:}T\ \exists B{:}INFO\_ELEMENT,\ \exists S2{:}SIGN$
$state(\gamma,t') \models prediction\_for(B,S2,A,S1) \wedge state(\gamma,t') \not\models holds\_in\_world(B,S2)\ ]$

IP18 Prediction consistency
If a certain prediction does not hold in the world, then its complement does hold.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT,\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \models prediction\_for(B,S2,A,S1)$
$\wedge\ state(\gamma,t) \not\models holds\_in\_world(B,S2)$
$\Rightarrow [\exists S3{:}SIGN \quad state(\gamma,t) \models holds\_in\_world(B,S3) \wedge S2 \neq S3\ ]$

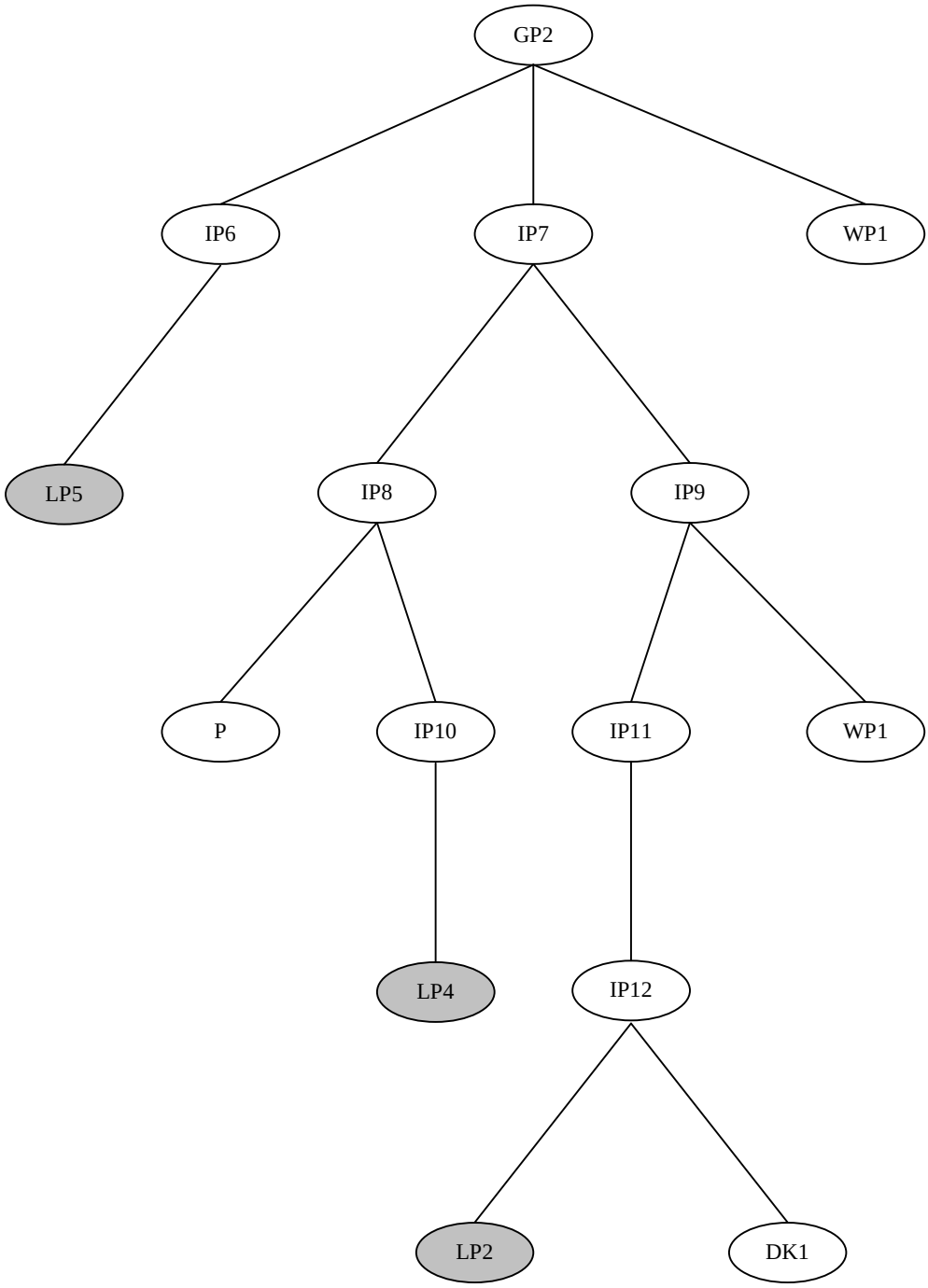
Appendix B. Logical Relationships between Dynamic Properties

This Appendix contains a number of trees of logical relationships relating global dynamic properties via intermediate dynamic properties to local dynamics properties. In particular, the following global dynamic properties have been worked out: GP1, GP2, and GP3. Here the grey ovals indicate that the 'grounding' variant of the property is used, which states that the conclusion derived by that particular property is unique.

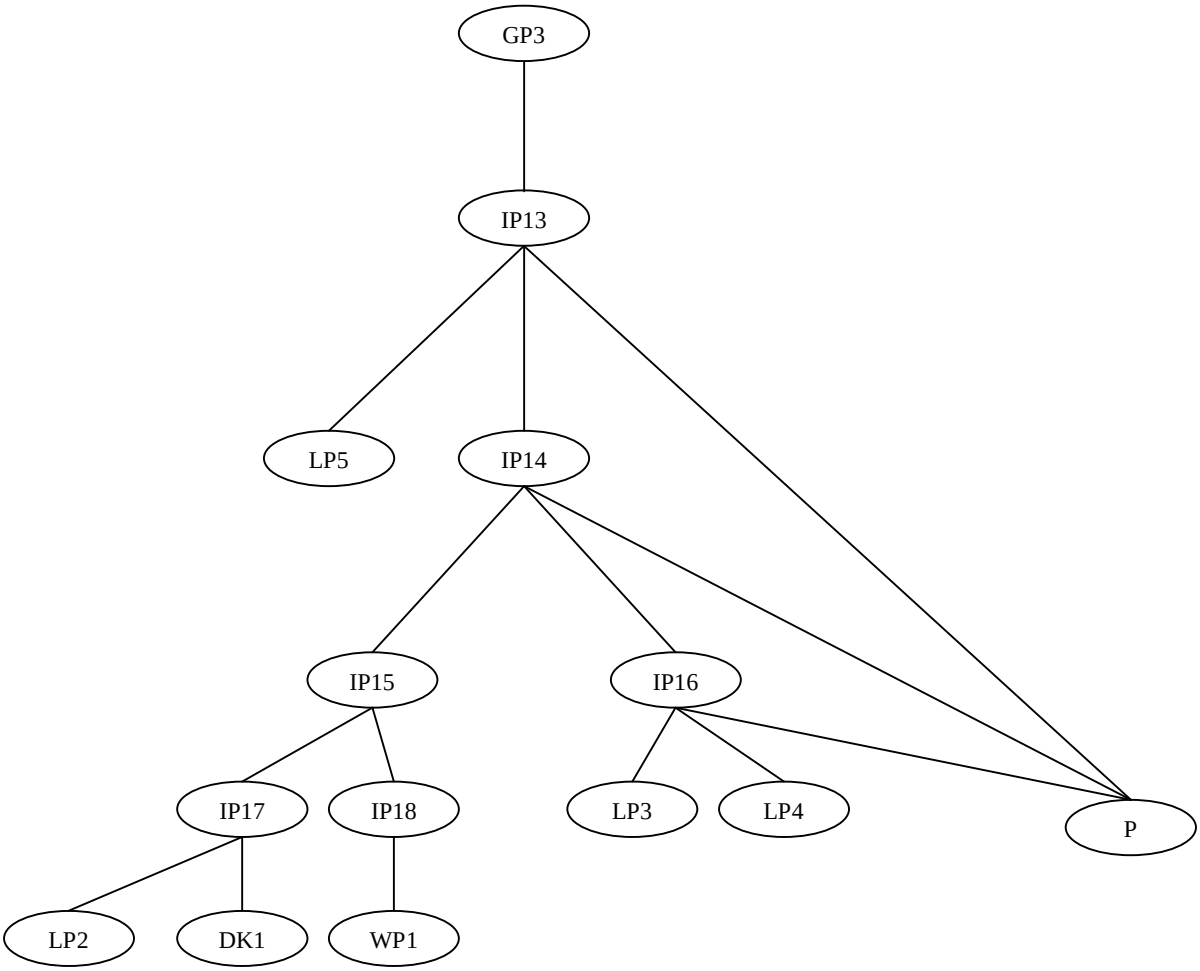
GP1



GP2



GP3



Appendix C. Transcripts

This Appendix contains two additional transcripts, and their formalisation. The left column contains the human transcript, the right column contains the formal counterpart. The complete set of transcripts of all human reasoning traces can be found at the following URL: http://www.cs.vu.nl/~tbosse/mastermind/human-traces.doc.

Example 1

Human transcript	Formalisation
All right, so I will try to say aloud as much as possible. *Yes, please.* So a black one means that one is in the right position, and two white ones that those are not in the right position. So all three colours are correct, because I already have three pins.	
Well, I just guess that white is in the right position...	focus_assumed(at(white, 2))
...and then I will swap the other two.	code_extension_for(code(blue, white, red), at(white, 2)) assumed(code(blue, white, red)) prediction_for(answer(black, black, black), code(blue, white, red))
[blue-white-red]	to_be_observed_for(answer, code(blue, white, red))
Okay. Why do you do this? Well, one of them is in the right position, so here I guessed one of them. And I know that these two colours are correct but that they are not in the right position so I have only those one as other possibility to change.	
Okay. Then my answer is very simple. That is already correct! *[black-black-black]*	observation_result_for(answer(black, black, black), code(blue, white, red))

Example 2

Human transcript	Formalisation
Ooh! Well, all those three are already in. So that is easy. So all those others are not part of it. Well, let's have a look, a black one, so one of the three is correct and the other two I should swap. So then I can either just continue until I have it, or make use of others, that is also possible. What shall I do? I will make use of others, I like that. Like this.	
So that one is correct, that's what I think for the moment.	focus_assumed(at(red, 1))
And then I put two yellow ones in it. And then I will look what it becomes.	code_extension_for(code(red, yellow, yellow), at(red, 1)) assumed(code(ed, yellow, yellow)) prediction_for(answer(black), code(red, yellow, yellow))
[red-yellow-yellow]	to_be_observed_for(answer, code(red, yellow, yellow))
Okay. Then the answer is like this... *[white]*	observation_result_for(answer(white), code(red, yellow, yellow))
Yes. So, I think then, the red one was not right in that position, so then it must have been one of the others.	rejected_code(code(red, yellow, yellow)) rejected_focus(at(red, 1))
So, now I will think, then it is for example the white one. That one was positioned correctly over there.	focus_assumed(at(white, 2))
But the red one was not placed correctly over there, so then the red one should be over there. Let's have a look, is... am I doing that right? Perhaps I make it extra difficult for myself, and then it is still not correct. Let's have a look, well, let's try that anyway. Then it should be like this and then it should be like this...	code_extension_for(code(blue, white, red), at(white, 2)) assumed(code(blue, white, red)) prediction_for(answer(black, black, black), code(blue, white, red))
[blue-white-red]	to_be_observed_for(answer, code(blue, white, red))
Okay. Then the answer is like this... *[black-black-black]* Yes! *Congratulations!*	observation_result_for(answer(black, black, black), code(blue, white, red))

References

Bosse, T., Jonker, C.M., and Treur, J. (2003). Simulation and analysis of controlled multi-representational reasoning processes. *Proc. of the Fifth International Conference on Cognitive Modelling, ICCM'03*. Universitats-Verlag Bamberg, 2003, pp. 27-32.

Bosse, T., Jonker, C.M., Schut, M.C., and Treur, J. (2004). Modelling Shared Extended Mind and Collective Representational Content. In: Bramer, M., Coenen, F., and Allen, T. (eds.), *Research and Development in Intelligent Systems XXI, Proceedings of AI-2004, the 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer Verlag, 2004, pp 19-32.

Braine, M.D.S., and O'Brien, D.P. (eds.) (1998). Mental Logic. Lawrence Erlbaum, London.

Brazier, F.M.T., Treur, J., Wijngaards, N.J.E., and Willems, M. (1999). Temporal semantics of compositional task models and problem solving methods. *Data and Knowledge Engineering*, vol. 29, 1998, pp. 17-42.

Byrne, R.M.J. (1989). Suppressing valid inferences with conditionals. *Cognition*, vol. 31, 1989, pp. 61-83.

Dardenne, A., Lamsweerde, A. van, and Fickas, S. (1993). Goal-directed Requirements Acquisition. *Science in Computer Programming*, vol. 20, pp. 3-50.

Dubois, E., Du Bois, P., and Zeippen, J.M. (1995). A Formal Requirements Engineering Method for Real-Time, Concurrent, and Distributed Systems. In: *Proceedings of the Real-Time Systems Conference, RTS'95*.

Fensel, D., Harmelen, F. van (1994). A comparison of languages which operationalize and formalize KADS models of expertise. *Knowledge Engineering Review*, Volume 9, pp. 105-146.

Herlea, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E. (1999). Specification of Behavioural Requirements within Compositional Multi-Agent System Design. In: F.J. Garijo, M. Boman (eds.), *Multi-Agent System Engineering, Proc. of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99*. Lecture Notes in AI, vol. 1647, Springer Verlag, 1999, pp. 8-27.

Johnson-Laird, P.N. (1983). Mental Models. Cambridge: Cambridge University Press.

Johnson-Laird, P.N., and Byrne, R.M.J. (1991). Deduction. Hillsdale, NJ:Erlbaum.

Jonker, C.M., and Treur, J. (1998). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), *Proceedings of the International Workshop on Compositionality, COMPOS'97*. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380. Extended version in: *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

Jonker, C.M., and Treur, J. (2002). Analysis of the Dynamics of Reasoning Using Multiple Representations. In: W.D. Gray and C.D. Schunn (eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci 2002*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2002, pp. 512-517.

Jonker, C.M., and Treur, J. (2003). Modelling the Dynamics of Reasoning Processes: Reasoning by Assumption. *Cognitive Systems Research Journal*. In press, 2003.

Kaufman, L., and Rousseeuw, P. (1990). Finding Groups in Data: an Introduction to Cluster Analysis, John Wiley and Sons, 1990.

Klahr, D., and Dunbar, K. (1988). Dual space search during scientific reasoning. Cognitive Science, 12(1), 1-55.

Knuth, D.E. (1977). *The Computer as Master Mind*. Journal of Recreational Mathematics, 9 (1976-77), 1-6.

Koyama, K., and Lai, T.W. (1994). *An Optimal Mastermind Strategy*. Journal of Recreational Mathematics, 1994.

Kowalski, R., and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4:67-95, 1986.

Nelson, T. (2000). *A Brief History of the Master Mind™ Board Game*. URL: http://www.tnelson.demon.co.uk/mastermind/history.html

Reiter, R. (2001). Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, 2001.

Rips, L.J. (1994). The Psychology of Proof: Deductive reasoning in human thinking. MIT Pres, Cambridge, Mass.

Schroyens, W. J., Schaeken, W., & d'Ydewalle, G. (2001). A meta-analytic review of conditional reasoning by model and/or rule: Mental models theory revised. Psychological report No. 278. University of Leuven. Laboratory of Experimental Psychology.

Simon H.A., and Lea, G. (1974). Problem solving and rule induction: A unified view. In L. Gregg (Ed.), Knowledge and Cognition (pp. 105-128). Hillsdale, NJ: Lawrence Erlbaum.

Stenning, K., and Lambalgen, M. van (2005). A working memory model of relations between interpretation and reasoning. *Cognitive Science Journal*, Elsevier Science Inc., Oxford, UK. In press.

Treur, J., and Wetter, Th. (eds.) (1993). Formal Specification of Complex Reasoning Systems, Ellis Horwood.

Yang, Y., and Bringsjord, S. (2001). Mental MetaLogic: a New Paradigm in Psychology of Reasoning. Extended abstract in: L. Chen, Y. Zhuo (eds.), Proc. of the Third International Conference on Cognitive Science, ICCS 2001. Beijing, pp. 199-204.

Yang, Y., and Johnson-Laird, P.N. (1999). A study of complex reasoning: The case GRE 'logical' problems. In M. A. Gernsbacher & S. J. Derry (Eds.) Proceedings of the Twenty First Annual Conference of the Cognitive Science Society, pp. 767-771.

PART II REASONING

CHAPTER 5

Requirements Analysis of an Agent's Reasoning Capability

This chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2005). Requirements Analysis of an Agent's Reasoning Capability. In: Henderson-Sellers, B. and Winikoff, M. (eds.), *Proceedings of the Seventh International Workshop on Agent-Oriented Information Systems, AOIS'05*, pp. 82-89.

Requirements Analysis of an Agent's Reasoning Capability

Tibor Bosse[1], Catholijn M. Jonker[2], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, treur}@cs.vu.nl
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl

Abstract. The aim of requirements analysis for an agent that is to be designed is to identify what characteristic capabilities the agent should have. One of the characteristics usually expected for intelligent agents is the capability of reasoning. This paper shows how a requirements analysis of an agent's reasoning capability can be made. Reasoning processes may involve dynamically introduced or retracted assumptions: 'reasoning by assumption'. It is shown for this type of reasoning how relevant dynamic properties at different levels of aggregation can be identified as requirements that characterise the reasoning capability. A software agent has been built that performs this type of reasoning. The dynamic properties have been expressed using the temporal trace language TTL and can and have been checked automatically for sample traces.

1. Introduction

Requirements analysis addresses the identification and specification of the functionality expected for the system to be developed, abstracting from the manner in which this functionality is realised in a design and implementation of this system; e.g., [9], [16], [21]. Recently, requirements analysis for concurrent systems and agent systems has been addressed in particular, for example, in [11], [13]. An agent-oriented view on requirements analysis can benefit from the more specific assumptions on structures and capabilities expected for agents, compared to software components in general. To obtain these benefits a dedicated agent-oriented requirements analysis process can be performed that takes into account specific agent-related structures and capabilities. For example, for a number of often occurring agent capabilities, a requirements analysis can be made and documented that is reusable in future agent-oriented software engineering processes. In the process of building agent systems, software engineering principles and techniques, such as scenario and requirements specification, verification, and validation, can be supported by the reusable results of such a requirements analysis.

In this paper the results are presented of a requirements analysis of an agent's reasoning capability. Since reasoning can take different forms, intelligent agents may sometimes require nontrivial reasoning capabilities. The more simple forms of reasoning amount to determining the deductive closure of a logical theory (a knowledge base), given a set of input facts. Requirements for such reasoning processes can be specified in the form of a functional relation between input and output states, abstracting from the time it takes to perform the reasoning; e.g., [22]. Properties of such a functional relation can be related to properties of a knowledge base used to realise the functionality, which provides possibilities for verification and validation of this knowledge; e.g., [18]. However, more sophisticated reasoning capabilities can better be considered as involving a process over

time; especially for nontrivial reasoning patterns the temporal aspects play an important role in their semantics; cf. [12], [19]. Therefore, within an agent-oriented software engineering approach to an agent's reasoning capability, requirements specification has to address dynamic properties of a reasoning process.

This paper shows how such a requirements analysis of the dynamics of an agent's reasoning capability can be made. The approach makes use of a semantic formalisation of reasoning processes by traces consisting of sequences of reasoning states over time, following the semantic formalisation introduced in [12]. Reasoning processes as performed by humans may involve dynamically introduced or retracted assumptions: a pattern used as a case study in this paper, further on called 'reasoning by assumption'. For requirements acquisition, it is to be shown for this type of reasoning which relevant dynamic properties can be identified that characterise the reasoning pattern.

A number of scenarios of practical human reasoning processes considered as 'reasoning by assumption' have been analysed and specified to identify requirements that are characteristic for this reasoning pattern. Required dynamic properties at different levels of aggregation (or grain size) have been identified. Logical relationships have been determined between dynamic properties at one aggregation level and those of a lower aggregation level. These characterising properties have been formalized using the temporal trace language TTL, thus enabling automated support of analysis. As an additional validation of this characterisation, a number of reasoning puzzles were used to acquire scenarios of further practical human reasoning processes that intuitively fit the pattern of reasoning by assumption [5]. Supported by software tools, the properties were checked against the formalised scenarios of these human traces, and confirmed.

The specified dynamic properties at the lowest aggregation level are in an executable format; they specify reasoning steps. Using a variant of Executable Temporal Logic [2], and a dedicated software environment for simulation that has been developed [3], these executable properties were used to generate simulation traces. Moreover, for these traces the (higher-level) dynamic properties were checked and confirmed, which validates the identified logical relationships between the dynamic properties at different aggregation levels.

Finally, a design of an existing software agent performing reasoning by assumption [15] was analysed. This agent was designed using the component-based design method DESIRE [6]. Using the DESIRE execution environment, for this agent a number of reasoning traces were generated. For these traces all identified dynamic properties (also the executable ones) were checked, and found confirmed.

In Section 2 the dynamic perspective on reasoning is discussed in some more detail, and focussed on the pattern 'reasoning by assumption'. Section 3 addresses some details of the language used. Section 4 presents a number of requirements in the form of dynamic properties identified for patterns of reasoning by assumption. Section 5 discusses relationships between dynamic properties at different aggregation levels. In Section 6 it is discussed in which respects verification has been performed. In Section 7 the contribution of the research presented in the paper is briefly discussed.

2. The Dynamics of Reasoning

Analysis of reasoning processes has been addressed from different areas and angles, for example, Cognitive Science, Philosophy and Logic, and AI. For reasoning processes in natural contexts, which are usually not restricted to simple deduction, dynamic aspects play an important role and have to be taken into account, such as dynamic focussing by posing goals for the reasoning, or making (additional) assumptions during the reasoning,

thus using a dynamic set of premises within the reasoning process. Also dynamically initiated additional observations or tests to verify assumptions may be part of a reasoning process. Decisions made during the process, for example, on which reasoning goal to pursue, or which assumptions to make, are an inherent part of such a reasoning process. Such reasoning processes or their outcomes cannot be understood, justified or explained without taking into account these dynamic aspects.

The approach to the semantical formalisation of the dynamics of reasoning exploited here is based on the concepts reasoning state, transitions and traces.

Reasoning state. A reasoning state formalises an intermediate state of a reasoning process. The set of all reasoning states is denoted by RS.

Transition of reasoning states. A transition of reasoning states or reasoning step is an element $< S, S' >$ of $RS \times RS$. A *reasoning transition relation* is a set of these transitions, or a relation on $RS \times RS$ that can be used to specify the allowed transitions.

Reasoning trace. Reasoning dynamics or reasoning behaviour is the result of successive transitions from one reasoning state to another. A time-indexed sequence of reasoning states is constructed over a given time frame (e.g., the natural numbers). Reasoning traces are sequences of reasoning states such that each pair of successive reasoning states in such a trace forms an allowed transition. A trace formalises one specific line of reasoning. A set of reasoning traces is a declarative description of the semantics of the behaviour of a reasoning process; each reasoning trace can be seen as one of the alternatives for the behaviour. In Section 3 a language is introduced in which it is possible to express dynamic properties of reasoning traces.

The specific reasoning pattern used in this paper to illustrate the approach is ‘reasoning by assumption’. This type of reasoning often occurs in practical reasoning; for example, in everyday reasoning, diagnostic reasoning based on causal knowledge, and reasoning based on natural deduction. An example of everyday reasoning by assumption is ‘Suppose I do not take my umbrella with me. Then, if it starts raining at 5 pm, I will get wet, which I don’t want. Therefore I'd better take my umbrella with me’. An example of diagnostic reasoning by assumption in the context of a car that won’t start is: ‘Suppose the battery is empty, then the lights won’t work. But if I try, the lights turn out to work. Therefore the battery is not empty.’ Examples of reasoning by assumption in natural deduction are as follows. Method of indirect proof: ‘If I assume A, then I can derive a contradiction. Therefore I can derive not A.’. Reasoning by cases: ‘If I assume A, I can derive C. If I assume B, I can also derive C. Therefore I can derive C from A or B.’.

Notice that in all of these examples, first a reasoning state is entered in which some fact is *assumed*. Next (possibly after some intermediate steps) a reasoning state is entered where *consequences* of this assumption have been *predicted*. Finally, a reasoning state is entered in which an *evaluation* has taken place; possibly in the next state the assumption is retracted, and conclusions of the whole process are added. In Section 3 and 4, this pattern is to be characterised by requirements.

3. Dynamic Properties

To specify properties on the dynamics of reasoning, the temporal trace language TTL used in [13] is adopted. This is a language in the family of languages to which also situation calculus [20], event calculus [17], and fluent calculus [14] belong.

Ontology. An ontology is a specification (in order-sorted logic) of a vocabulary. For the example reasoning pattern 'reasoning by assumption' the state ontology includes binary relations such as assumed, rejected, on sorts INFO_ELEMENT x SIGN and the relation prediction_for on INFO_ELEMENT x SIGN x INFO_ELEMENT x SIGN. Table 1 contains all relations that will be used in this paper, as well as their explanation. The sort INFO_ELEMENT includes specific domain statements such as car_starts, lights_burn, battery_empty, sparking_plugs_problem. The sort SIGN consists of the elements pos and neg.

Internal concepts:	
initial_assumption(A:INFO_ELEMENT, S:SIGN)	The agent believes that it is most plausible to assume (A,S). Therefore, this is the agent's default assumption. For example, if it is most likely that the battery is empty, this is indicated by initial_assumption(battery_empty, pos).
assumed(A:INFO_ELEMENT, S:SIGN)	The agent currently assumes (A,S).
prediction_for(A:INFO_ELEMENT, S1:SIGN, B:INFO_ELEMENT, S2:SIGN)	The agent predicts that if (B,S2) is true, then (A,S1) should also be true.
rejected(A:INFO_ELEMENT, S:SIGN)	The agent has rejected the assumption (A,S).
alternative_for(A:INFO_ELEMENT, S1:SIGN, B:INFO_ELEMENT, S2:SIGN)	The agent believes that (A,S1) is a good alternative assumption in case (B,S2) is rejected.
Input and output concepts:	
to_be_observed(A:INFO_ELEMENT)	The agent starts observing whether A is true.
observation_result(A:INFO_ELEMENT, S:SIGN)	If S is pos, then the agent observes that A is true. If S is neg, then the agent observes that A is false.
External concepts:	
domain_implies(A:INFO_ELEMENT, S1:SIGN, B:INFO_ELEMENT, S2:SIGN)	Under normal circumstances, (A,S1) leads to (B,S2). For example, an empty battery usually implies that the lights do not work.
holds_in_world(A:INFO_ELEMENT, S:SIGN)	If S is pos, then A is true in the world. If S is neg, then A is false.

Table 1. State ontology for the pattern 'reasoning by assumption'

Reasoning state. A (reasoning) state for ontology Ont is an assignment of truth-values {true, false} to the set of ground atoms At(Ont). The set of all possible states for ontology Ont is denoted by STATES(Ont). A part of the description of an example reasoning state s is:

assumed(battery_empty, pos) : true
prediction_for(lights_ burn, neg,battery_empty, pos) : true
observation_result(lights_burn, pos) : true
rejected(battery_empty, pos) : false

The standard satisfaction relation |== between states and state properties is used: S |== p means that state property p holds in state S. For example, in the reasoning state S above it holds S |== assumed(battery_empty, pos).

Reasoning trace. To describe dynamics, explicit reference is made to time in a formal manner. A fixed time frame T is assumed which is linearly ordered. Depending on the application, for example, it may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers). A trace γ over an ontology Ont and time frame T is a mapping $\gamma : T \rightarrow STATES(Ont)$, i.e., a sequence of reasoning states γ_t ($t \in T$) in STATES(Ont). The set of all traces over ontology Ont is denoted by $\Gamma(Ont)$, i.e., $\Gamma(Ont) = STATES(Ont)^T$. The set $\Gamma(Ont)$ is also denoted by Γ if no confusion is expected. Please note that in each trace, the current world state is included.

Expressing dynamic properties. States of a trace can be related to state properties via the formally defined satisfaction relation |== between states and formulae. Comparable to the approach in situation calculus, the sorted predicate logic temporal trace language TTL is built on atoms such as state(γ , t) |== p, referring to traces, time and state properties. This expression denotes that state property p is true in the state of trace γ at time point t. Here |== is a predicate symbol in the language (in infix notation), comparable to the Holds-predicate in situation calculus. Temporal formulae are built using the usual logical connectives and quantification (for example, over traces, time and state properties). The set TFOR(Ont) is the set of all temporal formulae that only make use of ontology Ont. We allow additional language elements as abbreviations of formulae of the temporal trace language. The fact that this language is formal allows for precise specification of dynamic properties. Moreover, editors can and actually have been developed to support specification of properties. Specified properties can be checked automatically against example traces to find out whether they hold.

Simulation. A simpler temporal language has been used to specify simulation models. This temporal language, the LEADSTO language [3], offers the possibility to model direct temporal dependencies between two state properties in successive states. This executable format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the LEADSTO language $\alpha \rightarrow\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a precise definition of the LEADSTO format, see [3]. A specification of dynamic properties in LEADSTO format has as advantages that it is executable and that it can easily be depicted graphically.

4. Dynamic Properties as Characterising Requirements

Careful analysis of the informal reasoning patterns discussed in Section 2 led to the identification of dynamic properties that can serve as requirements for the capability of reasoning by assumption. In this section a number of the most relevant of those properties are presented in both an informal and formal way. The dynamic properties identified are at three different levels of aggregation:

- **Local properties** address the step-by-step reasoning process of the agent. They represent specific transitions between states of the process: *reasoning steps*. These properties are represented in *executable* format, which means that they can be used to generate simulation traces.
- **Global properties** address the *overall* reasoning behaviour of the agent, not the step-by-step reasoning process of the agent. Some examples of global properties are presented, regarding matters as termination, correct reasoning, and result production.
- **Intermediate properties** are properties at an intermediate level of aggregation, which are used for the analysis of global properties (see also Section 5).

A number of local properties are given in Section 4.1. It will be shown how they can be used in order to generate simulation traces. Next, Section 4.2 provides some global properties, and Section 4.3 some intermediate properties.

4.1. Local Dynamic Properties

At the lowest level of aggregation, a number of dynamic properties have been identified for the process of reasoning by assumption. These local properties are given below (both in an informal and in formal LEADSTO notation):

LP1 (Assumption Initialisation)
The first local property LP1 expresses that a first assumption is made. Here, note that initial_assumption is an agent-specific predicate, which can be varied for different cases. Formalisation:
initial_assumption(A, S) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ assumed(A, S)

LP2 (Prediction Effectiveness)
Local property LP2 expresses that for each assumption that is made, all relevant predictions are generated. Formalisation:
assumed(A, S1) and domain_implies(A, S1, P, S2) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ prediction_for(P, S2, A, S1)

LP3 (Observation Initiation Effectiveness)
Local property LP3 expresses that all predictions made will be observed. Formalisation:
prediction_for(P, S1, A, S2) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ to_be_observed(P)

LP4 (Observation Result Effectiveness)
Local property LP4 expresses that, if an observation is made the appropriate observation result will be received. Formalisation:
to_be_observed(P) and holds_in_world(P, S) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ observation_result(P, S)

LP5 (Evaluation Effectiveness)
Local property LP5 expresses that, if an assumption was made and a related prediction is falsified by an observation result, then the assumption is rejected. Formalisation:
assumed(A, S1) and prediction_for(P, S2, A, S1) and observation_result(P, S3) and S2≠S3
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ rejected(A, S1)

LP6 (Assumption Effectiveness)
Local property LP6 expresses that, if an assumption is rejected, and there is still an alternative assumption available, this will be assumed. Formalisation:
assumed(A, S1) and rejected(A, S1) and alternative_for(B, S2, A, S1) and not rejected(B, S2)
$\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ assumed(B, S2)

LP7 (Assumption Persistence)
Local property LP7 expresses that assumptions persist as long as they are not rejected. Formalisation:
assumed(A, S) and not rejected(A, S) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ assumed(A, S)

LP8 (Rejection Persistence)
Local property LP8 expresses that rejections persist. Formalisation:
rejected(A, S) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ rejected(A, S)

LP9 (Observation Result Persistence)
Local property LP9 expresses that observation results persist. Formalisation:
observation_result(P, S) $\rightarrow\!\!\!\!\rightarrow_{0,0,1,1}$ observation_result(P, S)

Using the software environment that is described in [3], these local dynamic properties can be used to generate simulation traces. Using such traces, the requirements engineers and system designers obtain a concrete idea of the intended flow of events over time. A number of simulation traces have been created for several domains. An example simulation trace in the domain of car diagnosis is depicted in Figure 1. Here, time is on the horizontal axis, and the state properties and on the vertical axis. A dark box on top of the line indicates that the state property is true during that time period, and a lighter box below the line indicates that the state property is false. This figure shows the characteristic cyclic process of reasoning by assumption: making assumptions, predictions and observations for assumptions, then rejecting assumptions and creating new assumptions. As can be seen in Figure 1, it is first observed that the car does not start. On the basis of

this observation, an initial assumption is made that this is due to an empty battery. However, if this assumption turns out to be impossible (because the lights are burning), this assumption is rejected. Instead, a second assumption is made (there is a sparking plugs problem), which turns out to be correct.

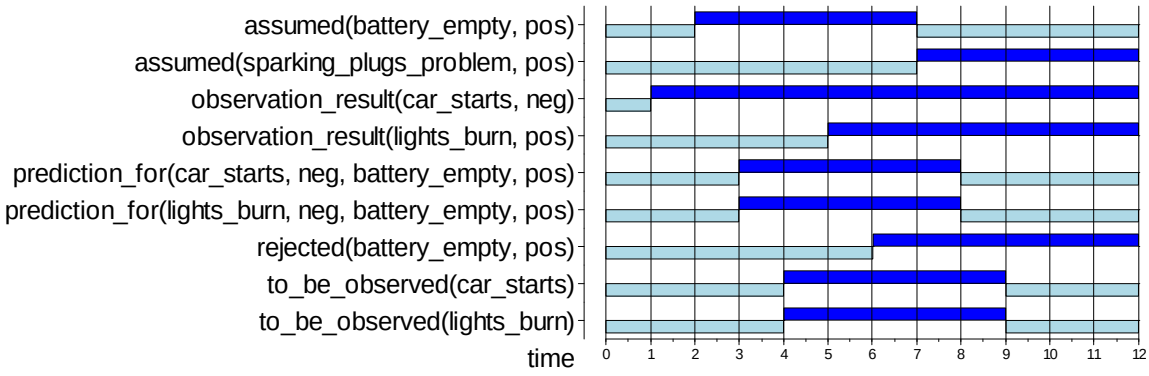


Figure 1. Example simulation trace

4.2. Global Dynamic Properties

At the highest level of aggregation, a number of dynamic properties have been identified for the overall reasoning process. These global properties are given below (both in an informal and in formal TTL notation):

GP1 (Reasoning Termination)
Eventually there is a time point at which the reasoning terminates.
$\forall\, \gamma : \Gamma\ \exists\, t{:}\ T$ termination(γ, t)

Here termination(γ, t) is defined as follows:
$\forall\, t'{:}\ T \quad t' \geq t \Rightarrow$ state(γ, t) = state(γ, t').

GP2 (Correctness of Rejection)
Everything that has been rejected does not hold in the world situation.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}\text{INFO\_ELEMENT}\ \forall S{:}\text{SIGN}$
state(γ,t) |== rejected(A,S) $\Rightarrow$
state(γ,t) |=/= holds_in_world(A,S)

GP3 (At least one not Rejected Assumption)
If the reasoning has terminated, then there is at least one assumption that has been evaluated and not rejected.
$\forall\, \gamma : \Gamma\ \forall\, t : T$ termination(γ, t)
$\Rightarrow$ [$\exists$ A: INFO_ELEMENT, $\exists$ S: SIGN
state(γ, t) |== assumed(A, S) $\wedge$ state(γ, t) |=/= rejected(A, S)]

In addition, some *assumptions on the domain* can be specified:

WP1 (Static World)
If something holds in the world, it will hold forever.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}\text{INFO\_ELEMENT}\ \forall S{:}\text{SIGN}$
state(γ,t) |== holds_in_world(A,S) $\Rightarrow$
[$\forall t'{:}T \geq t{:}T$ state(γ,t') |== holds_in_world(A,S)]
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}\text{INFO\_ELEMENT}\ \forall S{:}\text{SIGN}$
state(γ,t) |=/= holds_in_world(A,S) $\Rightarrow$
[$\forall t'{:}T \geq t{:}T$ state(γ,t') |=/= holds_in_world(A,S)]

WP2 (World Consistency)
If something holds in the world, then its complement does not hold.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \models holds\_in\_world(A,S1) \wedge S1 \neq S2 \Rightarrow$
$state(\gamma,t) \not\models holds\_in\_world(A,S2)$

DK1 (Domain Knowledge Correctness)
The domain-specific knowledge is correct in the world.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \models holds\_in\_world(A,S1) \wedge domain\_implies(A,S1,B,S2)$
$\Rightarrow state(\gamma,t) \models holds\_in\_world(B,S2)]$

4.3. Intermediate Dynamic Properties

In the sections above, on the one hand global properties for a reasoning process as a whole have been identified. On the other hand at the lowest level of aggregation local (executable) properties representing separate reasoning steps have been identified. It may be expected that any trace that satisfies the local properties automatically will satisfy the global properties (semantic entailment). As a form of verification it can be proven that the local properties indeed imply the global properties. To construct a transparent proof a number of *intermediate properties* have been identified. Examples of intermediate properties are property IP1 to IP7 shown below (both in an informal and in formal TTL notation).

IP1 (Proper Rejection Grounding)
If an assumption is rejected, then earlier on there was a prediction for it that did not match the corresponding observation result.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S1{:}SIGN$
$state(\gamma,t) \models rejected(A,S1) \Rightarrow$
$[\exists t'{:}T \leq t{:}T\ \exists B{:}INFO\_ELEMENT\ \exists S2,S3{:}SIGN$
$state(\gamma,t') \models prediction\_for(B,S2,A,S1) \wedge$
$state(\gamma,t') \models observation\_result(B,S3) \wedge S2 \neq S3]$

IP2 (Prediction-Observation Discrepancy implies Assumption Incorrectness)
If a prediction does not match the corresponding observation result, then the associated assumption does not hold in the world.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2,S3{:}SIGN$
$state(\gamma,t) \models prediction\_for(B,S2,A,S1) \wedge$
$state(\gamma,t) \models observation\_result(B,S3) \wedge S2 \neq S3 \Rightarrow$
$state(\gamma,t) \not\models holds\_in\_world(A,S1)$

IP3 (Observation Result Correctness)
Observation results obtained from the world indeed hold in the world.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \models observation\_result(A,S) \Rightarrow$
$state(\gamma,t) \models holds\_in\_world(A,S)$

IP4 (Incorrect Prediction implies Incorrect Assumption 1)
If a prediction does not match the facts from the world, then the associated assumption does not hold either.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2,S3{:}SIGN$
$state(\gamma,t) \models prediction\_for(B,S2,A,S1) \wedge$
$state(\gamma,t) \models holds\_in\_world(B,S3) \wedge S2 \neq S3 \Rightarrow$
$state(\gamma,t) \not\models holds\_in\_world(A,S1)$

IP5 (Observation Result Grounding)
If an observation has been obtained, then earlier on the corresponding fact held in the world.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A{:}INFO\_ELEMENT\ \forall S{:}SIGN$
$state(\gamma,t) \models\!\!= observation\_result(A,S) \Rightarrow$
$[\ \exists t'{:}T \leq t{:}T\ \ state(\gamma,t') \models\!\!= holds\_in\_world(A,S)\ ]$

IP6 (Incorrect Prediction implies Incorrect Assumption 2)
If a prediction does not hold in the world, then the associated assumption does not hold either.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \models\!\!= prediction\_for(B,S2,A,S1) \wedge$
$state(\gamma,t) \models\!\!/\!\!= holds\_in\_world(B,S2) \Rightarrow$
$state(\gamma,t) \models\!\!/\!\!= holds\_in\_world(A,S1)$

IP7 (Prediction Correctness)
If a prediction is made for an assumption that holds in the world, then the prediction also holds.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \models\!\!= prediction\_for(B,S2,A,S1) \wedge$
$state(\gamma,t) \models\!\!= holds\_in\_world(A,S1) \Rightarrow$
$state(\gamma,t) \models\!\!= holds\_in\_world(B,S2)$

5. Relationships Between Dynamic Properties

A number of logical relationships have been the identified between properties at different aggregation levels. An overview of all identified logical relationships relevant for GP2 is depicted as an AND-tree in Figure 2. Here the grey ovals indicate that the so-called *grounding* variant of the property is used. Grounding variants make a specification of local properties more complete by stating that there is no other means to produce certain behaviour. For example, the grounding variant of LP2 can be specified as follows (in TTL notation):

LP2G Prediction effectiveness groundedness
Each prediction is related (via domain knowledge) to an earlier made assumption.
$\forall\gamma{:}\Gamma\ \forall t{:}T\ \forall A,B{:}INFO\_ELEMENT\ \forall S1,S2{:}SIGN$
$state(\gamma,t) \models\!\!= prediction\_for(B,S2,A,S1) \Rightarrow$
$[\exists t'{:}T \leq t{:}T\ \ state(\gamma,t') \models\!\!= assumed(A,S1) \wedge$
$domain\_implies(A,S1,B,S2)]$

This property expresses that predictions made always have to be preceded by a state in which the assumption was made, and the domain knowledge implies the prediction.

The relationships depicted in Figure 2 should be interpreted as semantic entailment relationships. For example, the relationship at the highest level expresses that the implication IP1 & IP2 & WP1 => GP2 holds. A sketch of the proof for this implication is as follows.

> *Suppose IP1 holds. This means that, if an assumption is rejected at time t, then at a certain time point in the past (say t') there was a prediction for it that did not match the corresponding observation result. According to IP2, at the very same time point (t') the assumption for which the prediction was made did not hold in the world. Since the world is static (WP1), this assumption still does not hold at time point t. We may thus conclude that, if something is rejected at a certain time point, it does not hold in the world.*

Logical relationships between dynamic properties can be very useful in the analysis of empirical reasoning processes. For example, if a given person makes an incorrect rejection (i.e. property GP2 is not satisfied by the reasoning trace), then by a refutation process it can be concluded that either property IP1, property IP2, or property WP1 fails (or a combination of them). If, after checking these properties, it turns out that IP1 does

not hold, then this must be the case because LP5G does not hold. Thus, by this example refutation analysis it can be concluded that the cause of the unsatisfactory reasoning process can be found in LP5G. For more information about the analysis of human reasoning processes, see [5].

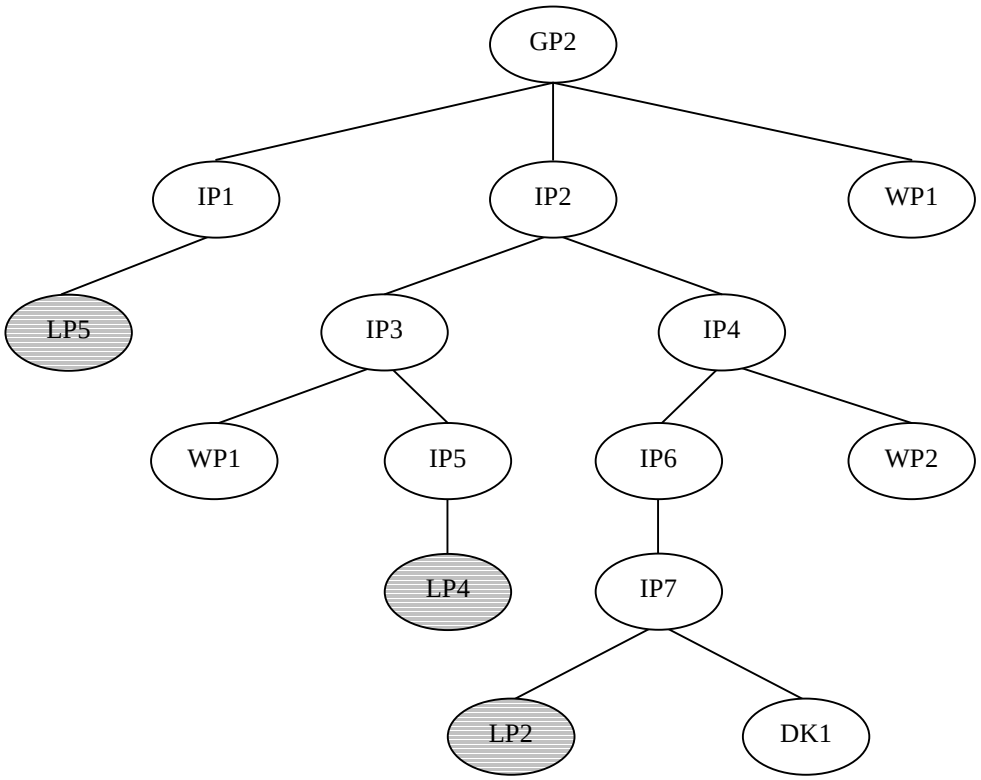


Figure 2. AND-Tree of Dynamic Properties

6. Verification

In addition to the simulation software environment described in Section 4, a special tool has been developed that takes a formally specified property and a set of traces as input, and verifies whether the property holds for the traces.

Using this checker tool, dynamic properties (of all levels) can be checked automatically against traces, irrespective of who/what produced those traces: humans, simulators or an implemented (prototype) system. A large number of such checks have indeed been performed for several case studies in reasoning by assumption. Table 2 presents an overview of all combinations of checks and their results. A '+' indicates that all properties were satisfied for the traces, a '+/-' indicates that some of the properties were satisfied.

	Human Traces (Taken from [5])	**Simulation Traces** (This paper)	**Prototype Traces** (Taken from [15])
Local Properties	+/-	+	+
Intermediate Properties	+/-	+	+
Global Properties	+/-	+	+

Table 2. Overview of the different verification results

As can be seen in Table 2, three types of traces were considered. First, the dynamic properties have been checked against human traces in reasoning experiments. It turned out that some of the properties were satisfied by all human traces, whereas some other properties sometimes failed. This implies that some properties are indeed characteristic for the pattern 'reasoning by assumption', whereas some other properties can be used to discriminate between different approaches to the reasoning. For example, human reasoners sometimes skip a step; therefore LP2 does not always hold. More details of these checks can be found in [5].

Second, the dynamic properties have been checked against simulation traces such as the one presented in Section 4.1 of this paper. As shown in Table 2, all properties eventually were satisfied for all traces. Note that this was initially not the case: in some cases small errors were made during the formalisation of the properties. Checking the properties against simulation traces turned out to be useful to localise such errors and thereby debug the formal dynamic properties.

Finally, all dynamic properties have been verified against traces generated by a prototype of a software agent performing reasoning by assumption, see [15]. This agent was designed on the basis of the component-based design method DESIRE, cf. [6]. Also for these traces eventually all dynamic properties turned out to hold.

To conclude, all automated checks described above have played an important role in the requirements analysis of reasoning capabilities of software agents, since they enabled the results of the requirements elicitation and specification phase to be formally verified and improved.

7. Discussion

In the literature, software engineering aspects of reasoning capabilities of intelligent agents have not been addressed well. Some literature is available on formal semantics of the dynamics of non-monotonic reasoning processes; for an overview, see [19]. However, these approaches focus on formal foundation and are far from the more practical software engineering aspects of actual agent system development.

In this paper it is shown how during an agent development process a requirements analysis can be incorporated. The desired functionality of the agent's reasoning capabilities can be identified (for example, in cooperation with stakeholders), using temporal specifications of scenarios and requirements specified in the form of (required) traces and dynamic properties. This paper shows for the example reasoning pattern 'reasoning by assumption', how relevant dynamic properties can be identified as requirements for the agent's reasoning behaviour, expressed using a temporal language, and verified and validated. Thus a set of requirements is obtained that is reusable in other agent development processes.

The language TTL used here allows for precise specification of these dynamic properties, covering both qualitative and quantitative aspects of states and their temporal relations. Moreover, software tools have been developed to (1) support specification of (executable) dynamic properties, and (2) automatically check specified dynamic properties against example traces to find out whether the properties hold for the traces. This provides a useful supporting software environment to evaluate reasoning scenarios both in terms of simulated traces (in the context of prototyping) and empirical traces (in the context of requirements elicitation and validation in co-operation with stakeholders). In the paper it is shown how this software environment can be used to automatically check the dynamic properties during a requirements analysis process. Note that it is not claimed that TTL is the only language appropriate for this. For example, most of the

properties encountered could as well have been expressed in a variant of linear time temporal logic. The language is only used as a vehicle; the contribution of the paper is in the method to requirements analysis of an agent's reasoning capability, and the reusable results obtained by that method.

For an elaborate description about the role that the current approach may take in Requirements Engineering, the reader is referred to [4]. In that paper, it is shown in detail how dynamic properties can be used to specify (both functional and non-functional) requirements of Agent Systems. Moreover, it is shown how these requirements may be refined and fulfilled according to the Generic Design Model (GDM) by Brazier *et al.* [6]. However, GDM is just one possible approach for Agent-Oriented Software Engineering. Recently, several other architectures have been proposed, for example, Tropos [8], KAOS [10] or GBRAM [1]. In future work, the possibilities may be explored to incorporate the approach based on dynamic properties presented here within such architectures. Especially for architectures that provide a specific language for formalisation of requirements (KAOS for example uses a real-time temporal logic to specify requirements in terms of goals, constraints and objects), these possibilities are promising.

References

[1] Antón, A.I. (1996). Goal-based Requirements Analysis, *Proc. of the International Conference on Requirements Engineering (ICRE'96)*, IEEE Computer Soc. Press, Colorado Springs, Colorado, USA, pp. 136- 144.

[2] Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M. (1996). *The Imperative Future: Principles of Executable Temporal Logic*, Research Studies Press Ltd. and John Wiley & Sons.

[3] Bosse, T., Jonker, C. M., van der Meij, L., and Treur, J. (2005). LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn (extended abstract). *Proc. of the 18th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, IEA/AIE 2005*. LNAI, Springer Verlag. In press.

[4] Bosse, T., Jonker, C.M., and Treur, J. (2004). Analysis of Design Process Dynamics. In: R. Lopez de Mantaras, L. Saitta (eds.), *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'04*, IOS Press, pp. 293-297.

[5] Bosse, T., Jonker, C.M., and Treur, J. (2005). Reasoning by Assumption: Formalisation and Analysis of Human Reasoning Traces. In: *Proceedings of the First International Work-conference on the Interplay between Natural and Artificial Computation, IWINAC'05*. LNAI, vol. 3561, Springer Verlag, pp. 430-439.

[6] Brazier, F.M.T., Jonker, C.M., and Treur, J. (2002). Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering*, vol. 41, pp. 1-28.

[7] Brazier F.M.T., Langen P.H.G. van, Treur J. (1996). A logical theory of design. In: J.S. Gero (ed.), *Advances in Formal Design Methods for CAD, Proc. of the Second International Workshop on Formal Methods in Design*. Chapman & Hall, New York, pp. 243-266.

[8] Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004). Tropos: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agent and Multi-Agent Systems*, vol. 8, pp. 203-236.

[9] Dardenne, A., Lamsweerde, A. van, and Fickas, S. (1993). Goal-directed Requirements Acquisition. *Science in Computer Programming*, vol. 20, pp. 3-50.

[10] Darimont, R., Delor, E., Massonet, P., and van Lamsweerde, A. (1998). GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering, *Proc. ICSE'98 - 20th International Conference on Software Engineering*, Kyoto, vol. 2, pp. 58-62.

[11] Dubois, E., Du Bois, P., and Zeippen, J.M. (1995). A Formal Requirements Engineering Method for Real-Time, Concurrent, and Distributed Systems. In: *Proceedings of the Real-Time Systems Conference, RTS'95*.

[12] Engelfriet, J., and Treur, J. (1995). Temporal Theories of Reasoning. *Journal of Applied Non-Classical Logics,* 5, pp. 239-261.
[13] Herlea, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E. (1999). Specification of Behavioural Requirements within Compositional Multi-Agent System Design. In: F.J. Garijo, M. Boman (eds.), *Multi-Agent System Engineering, Proc. of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99*. LNAI, vol. 1647, Springer Verlag, pp. 8-27.
[14] Hölldobler, S., and Thielscher, M. (1990). A new deductive approach to planning. *New Generation Computing*, 8:225-244.
[15] Jonker, C.M., and Treur, J. (2003). Modelling the Dynamics of Reasoning Processes: Reasoning by Assumption. *Cognitive Systems Research Journal*, vol. 4, pp. 119-136.
[16] Kontonya, G., and Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. John Wiley and Sons, New York.
[17] Kowalski, R., and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4, pp. 67-95.
[18] Leemans, N.E.M., Treur, J., and Willems, M. (2002). A Semantical Perspective on Verification of Knowledge. *Data and Knowledge Engineering*, vol. 40, pp. 33-70.
[19] Meyer, J.-J., Ch., and Treur, J. (eds.) (2001). *Dynamics and Management of Reasoning Processes*. Series in Defeasible Reasoning and Uncertainty Management Systems (D. Gabbay, Ph. Smets, series eds.), Kluwer Academic Publishers.
[20] Reiter, R. (2001). Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press.
[21] Sommerville, I., and Sawyer P. (1997). *Requirements Engineering: a good practice guide*. John Wiley & Sons, Chicester, England.
[22] Treur, J. (2002). Semantic Formalisation of Interactive Reasoning Functionality. *International Journal of Intelligent Systems*, vol. 17, pp. 645-686.

PART II REASONING

CHAPTER 6

Simulation and Analysis of Controlled Multi-Representational Reasoning Processes

Part of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2003). Simulation and Analysis of Controlled Multi-Representational Reasoning Processes. In: Detje, F., Doerner, D., and Schaub, H. (eds.), *Proceedings of the Fifth International Conference on Cognitive Modelling, ICCM'03*, Universitats-Verlag Bamberg, pp. 27-32.

Simulation and Analysis of Controlled Multi-Representational Reasoning Processes

Tibor Bosse[1], Catholijn M. Jonker[2], Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, the Netherlands
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, the Netherlands

Abstract. Multi-representational reasoning processes often show a variety of reasoning paths that can be followed. To analyse such reasoning processes with special attention for differences between individuals, it is required (1) to obtain an overview of the variety of different possibilities and (2) to address navigation and control within the reasoning process. This paper presents a simulation model and a formal analysis method for the dynamics of a controlled reasoning process in which multiple representations play a role. Reasoning Strategies to navigate through the space of possible reasoning states are modelled explicitly, and simulated. Simulation results are analysed by software tools on the basis of formalized dynamic properties. The variety of dynamic properties specified and the variety of traces simulated provides an overview for the individual differences between subjects that have been observed while solving multiplication problems.

1. Introduction

Human reasoning is often considered a process proceeding by accumulating a number of reasoning steps from start to end. An underlying assumption is that such a process can be analysed by studying each such step locally, in isolation from the rest of the reasoning process. Many reports of experimental research focus on one-trial-experiments where the number of reasoning steps is limited to one or, sometimes, at most two; e.g., (Rips, 1994; Johnson-Laird, 1983). However, a practical reasoning process often is not a straightforward accumulation of isolated steps. First, decisions to make a reasoning step may be not a local issue at the time point of the decision, but depend on the history and goals of the reasoning process as a whole. Second, often a multitude of reasoning paths is possible; only some of these actually reach the goal. Navigation and control in the sense of making a coherent set of choices at different time points to obtain one of the successful (and preferred according to one's own characteristics) paths is a nontrivial issue. Third, during the process steps may be taken that lead to a dead end, such that the reasoning process has to reconsider these steps, leading to revision of the reasoning path. These non-local aspects of a reasoning process require specific capabilities beyond, for example, the capability to locally apply modus ponens or modus tollens. Often some form of global reasoning planning and control is performed. Decisions to make or revise a specific reasoning step are made in the context of such a reasoning plan, which also has to be taken into account as part of a reasoning state.

In many cases the same information can be represented in different manners (e.g., in arithmetic, geometric or material form). Moreover, both internal (mental) and external (e.g., written or drawn) representations may play a role. The distinction between mental and external representations is also made in, e.g., (Hegarty, 2002). As the type of possible

reasoning steps may be different for different forms of representation, these differences of representation have to be accounted for in different reasoning states. In such cases the number of possible reasoning states is not very small, and, as a consequence, the number of possible reasoning paths, may be quite large. Coherent controlled navigation involving non-local aspects of decisions for reasoning steps is of major importance to deal with such a large number of possibilities.

This paper reports analysis and simulation of controlled multi-representation reasoning processes, in which the issues put forward play an important role. An analysis method for the dynamics of reasoning is based on formal definitions of possible reasoning states and traces, and dynamic properties of these traces are specified in the Temporal Trace Language TTL (Jonker and Treur, 1998; Herlea et al., 1999). This analysis method is supported by a software environment that is able to check traces against specified dynamic properties. For simulation the component-based agent design method DESIRE is used, cf. (Brazien et al., 2003). Traces generated by execution of a DESIRE model can be directly used as input of the analysis software environment.

In Section 2, the dynamic perspective on reasoning is discussed, with a focus on formalisation of the dynamics. Next, in Section 3, an example domain in reasoning with multiple representations is introduced. The example domain shows interaction between material, geometrical and arithmetical reasoning. It focuses on how to determine the outcome of multiplications such as 23 x 36, possibly using external arithmetic, geometric or material (based on Multi-base Arithmetic Blocks (MAB) material; e.g., Booker et al. 1997, English and Halford, 1995) representations. In Section 4, the design of the simulation model is presented. Various simulation traces have been generated, of which one example is briefly discussed. In Section 5, a number of dynamic properties for this type of reasoning are identified and formalised using TTL. Section 6 describes how these properties can be used to analyse existing (human or simulated) reasoning processes. Finally, in Section 7 the approach is summarised and the contribution of the research presented in the paper is discussed.

2. Formalising Reasoning Dynamics

Analysis of the cognitive capability to perform reasoning has been addressed from different areas and angles. Within Cognitive Science, the two dominant streams are the syntactic approach (based on inference rules applied to syntactic expressions, as common in logic), e.g., (Rips, 1994), and the semantic approach (based on construction of mental models); e.g., (Johnson-Laird, 1983; Yang and Johnson-Laird, 1999).

Reasoning steps in natural contexts are usually not restricted to the application of logical inference rules. For example, a step in a reasoning process may involve translation of information from one representation form (e.g., geometrical) into another one (e.g., arithmetical). Or, an additional assumption can be made, thus using a dynamic set of premises within the reasoning process. Decisions made at specific points in time during the process, for example, on which representations to use or which assumptions to make, are an inherent part of the reasoning. Such reasoning processes or their outcomes cannot be understood, justified or explained without taking into account these dynamic aspects.

To formalise the dynamics of a reasoning process, traces are used. *Reasoning traces* are time-indexed sequences of *reasoning states* over a time frame; for stepwise reasoning processes the set of natural numbers as a time frame is an appropriate choice. The set of all possible reasoning states defines the space where the reasoning takes place. Reasoning traces can be viewed as trajectories in this space, for which every (reasoning) step from one reasoning state to the next one is based on an *allowed transition*. If the possible

reasoning states and the allowed reasoning steps or transitions are characterised, the set of proper reasoning traces can be defined as the set of all possible sequences of reasoning states consisting only of allowed transitions.

2.1. Reasoning States

A *reasoning state* formalises an intermediate state of a reasoning process. The content of such a reasoning state usually can be analysed according to different aspects or dimensions. A reasoning state can include both internal (e.g., specific mental representations) and external elements (e.g., written or drawn notes). For example, part of the state may contain an external material representation, another part an external arithmetic representation, and yet another part an internal geometric representation. Furthermore, as pointed out in the Introduction, also control information has to be taken into account in a reasoning state. Accordingly, the reasoning state is structured as a composition of (i.e., a tuple of) a number of parts, indexed by some set I. This index set includes different aspects or views taken on the state, e.g., I is the set

{control, extmaterial, extgeometric, extarithmetic, intmaterial, intgeometric, intarithmetic}.

The set of reasoning states RS can be characterised as a Cartesian product $RS = \Pi_{i \in I} RS_i$ where RS_i is the set of all states for the aspect indicated by i. For example, $RS_{extgeometric}$ may denote the set of all possible external (drawn) geometric representations. This Cartesian product formalises the multi-dimensional space where the reasoning takes place. For a reasoning state, which is a vector $S = (S_i)_{i \in I} \in RS$ in this space, the S_i are called its *parts*.

2.2. Reasoning Steps

A transition from one reasoning state to another reasoning state, i.e., an element $< S, S' >$ of $RS \times RS$, formalises one *reasoning step*; sometimes also denoted by $S \rightarrow S'$. Transitions differ in the set of parts that are involved. The most complex transitions change all parts of the state in one step. However, within stepwise reasoning processes, usually transitions only involve a limited number of parts of the state, e.g., one to three. In the current approach we concentrate on this class of transition types.

For example, when a modification in the reasoning state is made solely within an internal geometric representation, only the internal geometric part of the state changes (geometric reasoning step):

$$intgeometric \rightarrow intgeometric$$

Other types of transitions involve more than one part. For example, if an external geometric representation is extended on the basis of an internal geometric representation, then two parts of the state are involved: the external geometric arithmetic part and the internal geometric part:

$$extgeometric \times intgeometric \rightarrow extgeometric$$

(e.g., the external geometric representation is extended or modified with results from the internal geometric representation)

If control information is incorporated in the modelling approach the number of involved parts is even higher, since every transition involves the control part; e.g.:

$$extgeometric \times intgeometric \times control \rightarrow extgeometric$$

(e.g., the external geometric representation is extended or modified with results from the internal geometric representation and some control information)

2.3. Reasoning Traces

Reasoning dynamics results from successive reasoning steps, i.e., successive transitions from one reasoning state to another. Thus a *reasoning trace* is constructed: a time-indexed sequence of reasoning states γ_t ($t \in T$), where T is the time frame used (the natural numbers). A reasoning trace can be viewed as a trajectory in the multi-dimensional space $RS = \Pi_{i \in I} RS_i$ of reasoning states. An example of such a reasoning trace will be discussed in Section 3.2. Reasoning traces are sequences of reasoning states subject to the constraint that each pair of successive reasoning states in this trace forms an allowed transition. A trace formalises one specific line of reasoning.

3. Example Domain: Multiplication

In this section, an example domain in multi-representation reasoning is used to illustrate the approach put forward: how to determine the outcome of multiplications such as 23 x 36. When solving such multiplications, human may use multiple different representations in their reasoning, depending on the approach used during the education. This example focuses on the interaction between arithmetical, geometrical and material reasoning. Experiences on using such processes with children (8-9 years old) in class rooms have been reported, e.g., by Dekker et al. (1982), see also (Hutton, 1977). Also teaching quadratic equations can be supported by such visualisations as discussed, e.g., by Bruner (1968), pp. 59-63. For further explorations of the idea to use visualisations in pre-algebraic reasoning, see (Koedinger and Terao, 2002).

3.1. Basic Skills

For the example domain, a number of basic skills have been identified, that can be used within the reasoning. In terms reasoning steps (as discussed in the previous section), these basic skills consist of three types of one-component transitions of reasoning states, and four transition types involving two components:

- *arithmetical* reasoning steps: arithmetic $\rightarrow$ arithmetic
- *geometrical* reasoning steps: geometric $\rightarrow$ geometric
- *material* reasoning steps: material $\rightarrow$ material
- *translations* of an arithmetical representation into a geometrical representation:
  geometric x arithmetic $\rightarrow$ geometric
- *translations* of a geometrical representation into an arithmetical representation:
  arithmetic x geometric $\rightarrow$ arithmetic
- *translations* of an arithmetical representation into a material representation:
  material x arithmetic $\rightarrow$ material
- *translations* of a material representation into an arithmetical representation:
  arithmetic x material $\rightarrow$ arithmetic

The idea is that more experienced reasoners possess more basic skills than less experienced reasoning. Less experienced reasoners require only simple arithmetical steps. They can perform the more complicated steps via the geometrical or material representation. The skills can be defined (informally) in the form of the following transitions:

A. *Arithmetic skills* (arithmetic → arithmetic)
bs7. splitting a number in 'tens' and single digits: 23 = 20 + 3
bs8. translating a multiplication of two complex number to the multiplication of the two sums of a 'ten' and a single digit:: 23x36 = (20+3)x(30+6)
bs9. multiplication of two numbers starting with a nonzero digit, followed by zero or more zeros, such as 20x8, 60x30.
bs10. applying the distribution law: (20+3)x(30+6) = (20x30)+(20x6)+(3x30)+(3x6)
bs11. extracting partial multiplication problems from a complex expression: (20x30)+(20x6)+(3x30)+(3x6) => (20x30)
bs12. filling in the solution to a partial multiplication problem in a complex expression
bs13. addition of a list of numbers of up to 4 digits, such as 600 + 120 + 90 + 18
bs14. concluding that the solution of the addition is the solution of the initial multiplication problem

B. *Geometric skills* (geometric → geometric)
bs4. partitioning a rectangle in non-overlapping areas based on partitionings of its sides

C. *Material skills* (material → material)
bs19. placing blocks inside the frame of a rectangle

D. *Translation skills* (geometric x arithmetic → geometric)
bs1. drawing a rectangle with arithmetically given dimensions
bs2. partitioning a line segment according to a splitting of its length
bs3. determining the surface of a rectangle from the multiplication of the lengths of its sides

E. *Translation skills* (arithmetic x geometric → arithmetic)
bs5. translating the area of a rectangle into the multiplication of the lengths of its sides
bs6. translating the area of a combination of nonoverlapping areas into the sum of the areas

F. *Translation skills* (material x arithmetic → material)
bs15. building the frame of a rectangle with arithmetically given dimensions
bs16. determining the surface of a group of identical blocks from the multiplication of the amount of blocks and the area of an individual block

G. *Translation skills* (arithmetic x material → arithmetic)
bs20. translating the area of a group of identical blocks into the multiplication of the amount of blocks and the area of an individual block
bs23. translating the area of a combination of groups of different block sizes into the sum of the areas of the groups

Notice that, in this notation no difference is made between the internal and the external elements of the reasoning states. However, the skills can easily be extended with this information. For example, basic skill bs1 can be extended in the two following ways:

bs1'. drawing a rectangle with arithmetically given dimensions on a piece of paper (intgeometric x intarithmetic → extgeometric)
bs1". imagining a rectangle with arithmetically given dimensions (intgeometric x intarithmetic → intgeometric)

A variety of (part of the) possible reasoning paths determined by these transitions is depicted in a simplified manner in Figure 1. For the sake of simplicity transitions between

geometric and material representations have been left out. The numbers refer to basic skills. The boxes refer to (part of) the reasoning states. For example, the transition labelled "4" refers to skill bs4, i.e., partitioning a rectangle in non-overlapping areas, based on a partitioning of its sides, and the transitions labelled "7" refer to skill bs7, i.e., splitting a number in tens and digits.

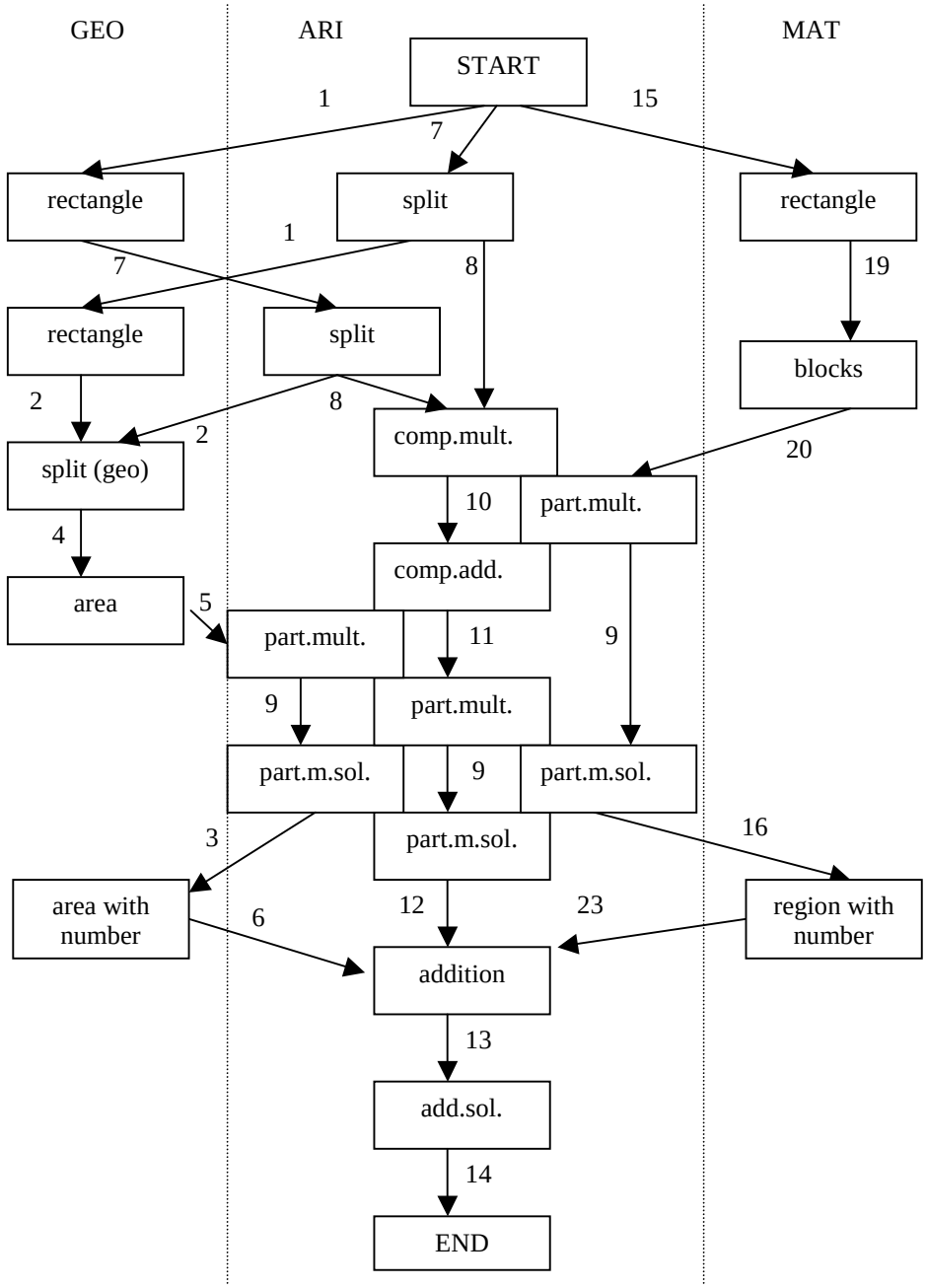


Figure 1. Variety of reasoning paths

3.2. Example Multi-Representational Reasoning Process

To illustrate the idea of the basic skills, Figure 2 presents a detailed reasoning trace. The starting problem for this trace was the following: "What is the outcome of the

multiplication 23 x 36?" In this example, only geometrical and arithmetical representations are used. The example corresponds to a particular navigation path through Figure 1.

Step 1 bs1 representation translation
Create a rectangle of 23 x 36.

Step 2 bs7 arithmetic reasoning
Split the numbers into the 'tens' and single digits: 23 = 20 + 3; 36 = 30 + 6

Step 3 bs2 representation translation
Translation of the arithmetical splitting of the numbers into partitions of the sides within the geometrical representation.

Step 4 bs4 geometric reasoning
Partition the area of the rectangle according to the partitioning of the sides.

Step 5 bs5 representation translation
For each part identify the corresponding arithmetical expression for its area: 20 x 30, 20 x 6, 3 x 30, 3 x 6

Step 6 bs9 arithmetic reasoning
Determine the outcomes of the four multiplications
20 x 30 = 600; 20 x 6 = 120;
3 x 30 = 90; 3 x 6 = 18

Step 7 bs3 representation translation
Identify the areas of the parts of the rectangle based on the outcomes of the multiplications.

Step 8 bs6 representation translation
Identify the corresponding arithmetical relation:
600 + 120 + 90 + 18.

Step 9 bs13 arithmetic reasoning
Calculate the sum: 600 + 120 + 90 + 18 = 828.

Step 10 bs14 arithmetic reasoning
Conclude that this is the solution: 828.

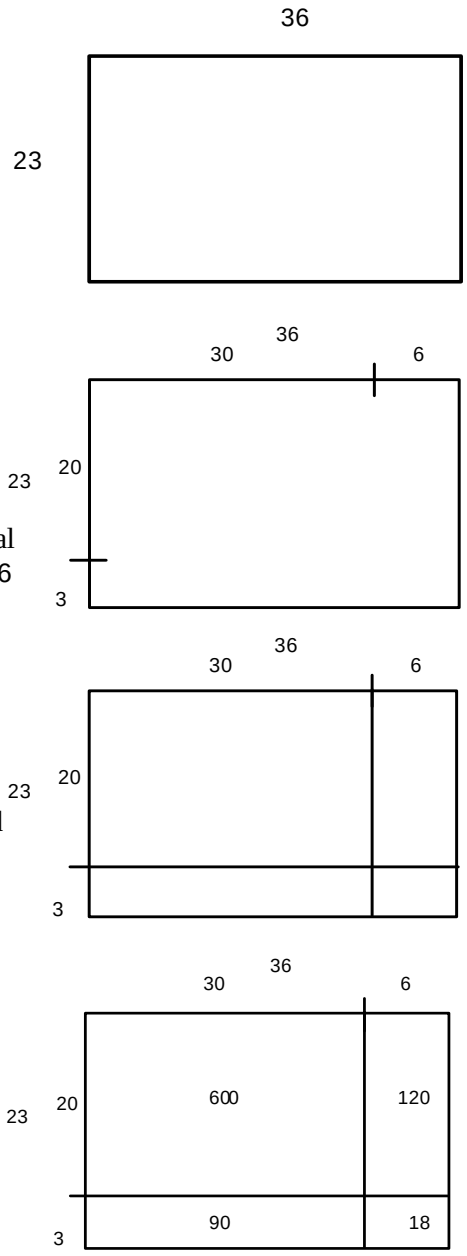


Figure 2. Example reasoning trace

4. Simulation Model

The simulation model[5] is based on agent modelling techniques, in particular the component-based agent modelling approach DESIRE; cf. (Brazier et al., 2002). At the highest level of abstraction, two components play a role in the system, i.e., the reasoning agent (called *Alan*) and the *External World*. Figure 3 depicts an overview of the components of the simulation model.

Alan can perform actions and observations, executed in the external world, and receive observation results as input from the external world. After Alan generates a certain action to be performed (e.g., draw a rectangle with sides 23 x 36), this action is transferred to the external world and executed there. The result of the action (e.g., a rectangle with corners A, B, C, D and sides 23 x 36 drawn on a piece of paper) will occur, with a certain delay, within the external world. Thus, the execution of physical actions by the agent is modelled as part of the component external world. Several kinds of physical actions are involved: writing things down (e.g., numbers), drawing pictures, and placing objects (e.g., blocks). Besides performing actions, Alan can pro-actively observe the world. The agent does this by explicitly determining what aspects of the world it is interested in: its observation focus. This focus is then transferred to the external world, which in return provides the corresponding observation result.

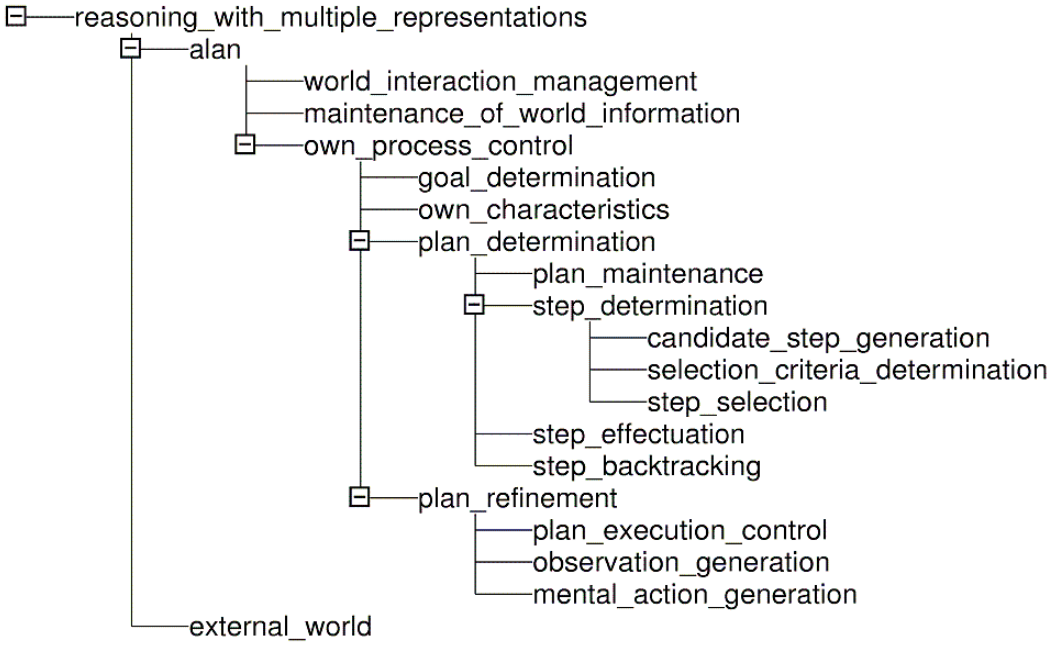


Figure 3. Overview of the components of the simulation model

4.1. Reasoning Agent

The approach used in this paper assumes that for every action a mental and a physical part can be distinguished and modelled (e.g., imagining a rectangle with sides 23 x 36 vs. actually drawing such a rectangle). Whilst the external world is concerned with the physical parts of the actions, everything that is represented within the agent is mental. To

[5] A complete specification of the model (with clickable components) can be found at www.cs.vu.nl/~wai/GTM/rmr/.

be able to make a clear distinction between the two concepts, a different notation is used for both types of information, e.g., rectangle(A, B, C, D, 23, 36) denotes a specific rectangle in the world, whereas entity(shape([]), parameters(23, 36)) denotes the internal representation. Internal representations can be created on the basis of an observation, but also on the basis of internal reasoning.

The composition of the reasoning agent Alan is based on the generic agent model as described in (Brazier et al., 2002). Three of the generic agent components are used in our model, namely *World Interaction Management, Maintenance of World Information* and *Own Process Control*. The other generic agent components were not needed within this model.

4.1.1. World Interaction Management

The component *World Interaction Management* handles the interaction with the external world, i.e., observation and action performance. It interprets information from the world and makes it available for relevant other components. Also it prepares information on actions to be performed.

4.1.2. Maintenance of World Information

The task of the component *Maintenance of World Information* is to maintain a (partial) world model, i.e., a snapshot of the present world state. In this domain, this world model is restricted to the observed information about objects that the agent has manipulated itself, such as the numbers it has written down. Moreover, since the agent does not necessarily have to perform each intermediate step physically, some imaginary world model must be maintained as well. This model describes the world like it would be after the physical execution of some steps, without these steps actually being performed. As both models contain information about a (possible) state of the world, both are maintained by Maintenance of World Information. An important issue is the amount of time that the world models persist within the component. In the current model, this duration is very short; thus, the component can be compared with part of the short-term memory: the information enters, is (possibly) used by another component, and very quickly disappears. Hence, whenever the information is needed later on, it has to be created again (either by observation or by imagination). This loss of information is modelled by clearing the contents of the component soon after it has entered. However, the duration of this period can easily be modified.

4.1.3. Own Process Control

According to the generic agent model, tasks of the component *Own Process Control* are the processes the agent uses to control its own activities (e.g., determining, monitoring and evaluating its own goals and plans), but also the processes of maintaining a self model. The way the tasks are performed is described in detail in the next section.

4.2. Own Process Control

Own Process Control consists of four sub-components: *Goal Determination*, *Own Characteristics*, *Plan Determination* and *Plan Refinement*, see Figure 3. These components are responsible for, respectively, determining the agent's goals, its own characteristics, planning the reasoning process at an abstract level, and actually performing the reasoning process. Their exact working is described in this section.

4.2.1. Goal Determination

For the application in question, *Goal Determination* is a relatively simple component. It contains information about the initial multiplication problem the agent desires to solve. The fact that the initial problem is represented here reflects the situation that the desire to solve this particular problem has popped up within the agent's mind spontaneously. However, in many cases the determination of goals is a more complex process. Therefore, the component can easily be extended to simulate a more dynamic form of goal determination (e.g., involving the possibility to modify and drop goals).

4.2.2. Own Characteristics

The component *Own Characteristics* contains a self-model, which includes several aspects. In the first place, it includes (self-)information on the basic skills that the reasoning agent thinks to possess. Note that this does not necessarily mean that the agent indeed has all these skills. For instance, it is well possible that the agent believes to be able to apply the distribution law of arithmetic, whilst during execution it turns out that it does not (i.e., the agent overestimated itself). Also, the opposite is possible. In that case, an agent possesses certain skills of which it does not know it has them. As a consequence, it will never use these skills. In the case of the over confident agent, when a certain skill has failed (i.e., the agent planned to use it, but at the end, it could not), Own Characteristics revises the self-knowledge of the agent by asserting that it does not have the skill after all. Second, Own Characteristics is used to store the agent's profile with respect to its problem solving strategy for the multiplication problem. Two aspects are represented: (1) a list of priorities among the different representations that can be used while solving the problem (e.g., the profile ari-geo-mat indicates that the agent prefers arithmetical representations to geometrical and material ones), and (2) to what extent steps in the reasoning process have to be performed physically. This way, several types of agents can be modelled, varying from those that write down every step to those that write down nothing. As a final remark, notice that, although DESIRE offers the opportunity to dynamically add changes in the specification (and thereby realise an open state space), this has not been done within the current model.

4.2.3. Plan Determination

Before actually solving the problem, the reasoning agent makes an abstract plan (e.g., a particular navigation route through Figure 1). *Plan Determination* is responsible for this planning process. Its input consists of the agent's own goal and characteristics. Based on this information, and knowledge about pre- and postconditions of the basic skills, Plan Determination explores the entire reasoning process at an abstract level. It uses abstract knowledge about when a certain basic skill can be applied (preconditions), and what the effect of this application will be (postconditions). The pre- and postconditions are expressed in an abstract way; e.g., they do not contain any numbers. While planning, Plan Determination continuously matches the current state of the explored plan against the preconditions of all basic skills, in order to determine which skills are applicable. It then uses its strategy profile in order to select one of the applicable skills. Subsequently, the skill is evaluated by adding its (abstract) postcondition to the current state of the explored plan. This way, the component constructs a complete list of steps to be performed, that would solve the multiplication problem. Furthermore, the component uses backtracking in situations where no more basic skills are applicable. Finally, if no solution can be found at all, this is also indicated. The sub-components of Plan Determination will be described in Section 4.3.

4.2.4. Plan Refinement

Abstract plans, generated by Plan Determination, are transferred to the component *Plan Refinement*. This component, which consists of the sub-components *Plan Execution Control*, *Precondition Acquisition Initiation* and *Mental Action Execution*, is responsible for the refinement of the basic steps, i.e., it determines the specific mental and physical actions associated to a basic step of the abstract plan (e.g., it refines bs4 to bs4m). Moreover, it executes the detailed mental actions associated to the basic steps. This is done by repeating the following activities. First, Plan Execution Control selects the first step of the (remaining) plan to be executed. Second, Precondition Acquisition Initiation determines what observations have to be made to provide the agent with the necessary information for the application of the selected step. For instance, if the selected step is to draw a rectangle, it is important to know the dimensions of the rectangle. Third, as soon as this information has been obtained, Mental Action Execution creates a mental image of the result of the application of the mental action (with instantiated variables, e.g., 'a rectangle with sides 23 x 36', denoted by entity(shape([]), parameters(23, 36))). This mental image is then stored within Maintenance of World Information. After that, Plan Execution Control decides whether to perform the associated physical action as well, depending on the agent's own characteristics. Then, the physical action either is or is not executed (within the External World component), after which the next step of the plan is treated by Plan Execution Control. Finally, if the agent is unable to perform an action that it had planned to do because it lacks either the mental or the physical skill for that action, notification with the name of the skill that failed is transferred to Own Characteristics. As a consequence, this latter component will revise its self-model, so that Plan Determination can construct a new plan more adequately.

4.3. Plan Determination

Plan Determination consists of the components *Plan Maintenance*, *Step Determination*, *Step Effectuation* and *Step Backtracking*. Plan Maintenance keeps track of all kinds of information concerning the 'current' state of the explored reasoning process, such as the (abstract) steps that have been applied successfully, those that have failed and those that have not been applied yet. Step Determination determines the next step to be added to the current plan in three phases. First, it determines which steps are currently applicable, by matching the preconditions of abstract steps against the current state of the exploration. Second, based on the applicable steps and the agent's strategy profile, it decides whether it will make an arithmetic, geometric, or material step. And third, based on the chosen representation, it will select one single step. The components responsible for the three phases are called, respectively, *Candidate Step Generation*, *Selection Criteria Determination*, and *Step Selection*. Finally, the selected step is passed to Step Effectuation. However, if, independently of the representation, no steps are applicable, this failure is indicated, so that the backtracking component can become active. Step Effectuation explores the execution of the selected abstract step by adding the postcondition of the step to the current state of the simulation. Step Backtracking becomes active whenever no more steps are applicable and uses a standard backtracking algorithm.

4.4. Example Simulation Trace

Using the model described above, a number of simulations have been performed. An example of a resulting simulation trace is shown in Table 1. In this trace, both geometric

and arithmetic skills are used to solve the problem, although there is a preference for the geometric skills. More simulation traces are included in Appendix A.

Step	Information Derived
0	strategy profile: geo-ari-mat
0	available abstract skills: all skills
0	available mental skills: all skills
0	available physical skills: all skills
0	represent physically: all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))
2	plan([bs1, bs7, bs2, bs4, bs5, bs9, bs3, bs6, bs13, bs14])
3	is_represented_in_world(arithmetic, multiplication(23, 36))
4	mental_representation(geometric, entity(shape("[]"), parameters(23, 36)))
5	is_represented_in_world(geometric, rectangle('A', 'B', 'C', 'D', 23, 36))
6	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(36, 30, 6)))
6	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(23, 20, 3)))
7	is_represented_in_world(arithmetic, split(36, 30, 6))
7	is_represented_in_world(arithmetic, split(23, 20, 3))
8	mental_representation(geometric, entity(shape("-"), name('A', 'B'), parameters(20, 3)))
8	mental_representation(geometric, entity(shape("-"), name('A', 'D'), parameters(30, 6)))
9	is_represented_in_world(geometric, split('A', 'B', 20, 3))
9	is_represented_in_world(geometric, split('A', 'D', 30, 6))
10	mental_representation(geometric, entity(shape("[]"), name('A11'), parameters(20, 30)))
10	mental_representation(geometric, entity(shape("[]"), name('A12'), parameters(20, 6)))
10	mental_representation(geometric, entity(shape("[]"), name('A21'), parameters(3, 30)))
10	mental_representation(geometric, entity(shape("[]"), name('A22'), parameters(3, 6)))
11	is_represented_in_world(geometric, area('A11', 20, 30))
11	is_represented_in_world(geometric, area('A12', 20, 6))
11	is_represented_in_world(geometric, area('A21', 3, 30))
11	is_represented_in_world(geometric, area('A22', 3, 6))
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 6)))
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 30)))
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 6)))
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 30)))
13	is_represented_in_world(arithmetic, partial_multiplication('A11', 20, 30))
13	is_represented_in_world(arithmetic, partial_multiplication('A12', 20, 6))
13	is_represented_in_world(arithmetic, partial_multiplication('A21', 3, 30))
13	is_represented_in_world(arithmetic, partial_multiplication('A22', 3, 6))
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 6, 18)))
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 30, 90)))
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 6, 120)))
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 30, 600)))
15	is_represented_in_world(arithmetic, multiplication_solution(3, 6, 18))
15	is_represented_in_world(arithmetic, multiplication_solution(3, 30, 90))
15	is_represented_in_world(arithmetic, multiplication_solution(20, 6, 120))
15	is_represented_in_world(arithmetic, multiplication_solution(20, 30, 600))
16	mental_representation(geometric, entity(shape("[]"), name('A11'), area_with_number(600)))
16	mental_representation(geometric, entity(shape("[]"), name('A12'), area_with_number(120)))
16	mental_representation(geometric, entity(shape("[]"), name('A21'), area_with_number(90)))
16	mental_representation(geometric, entity(shape("[]"), name('A22'), area_with_number(18)))
17	is_represented_in_world(geometric, area_with_number('A11', 600))
17	is_represented_in_world(geometric, area_with_number('A12', 120))
17	is_represented_in_world(geometric, area_with_number('A21', 90))
17	is_represented_in_world(geometric, area_with_number('A22', 18))
18	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(600, 120, 90, 18)))
19	is_represented_in_world(arithmetic, addition(600, 120, 90, 18))
20	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(600, 120, 90, 18, 828)))
21	is_represented_in_world(arithmetic, addition_solution(600, 120, 90, 18, 828))
22	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))
23	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))

Table 1. Example simulation trace

As can be seen in the table, the trace first contains a description of the characteristics of the agent (step 0), then the arithmetic problem is mentally represented (step 1) and the abstract plan is produced (step 2). Due to the strategy profile of the agent, the plan shows as many basic geometric skills as possible (this corresponds to a route through the left part

of Figure 1). Every step is represented both mentally and physically, corresponding to the agent's characteristics. Since the agent has all skills both in abstracto and in concreto, no backtracking was necessary either during plan determination or plan execution.

5. Dynamic Properties

To specify properties on the dynamics of a reasoning process, the temporal trace language TTL used by Herlea et al. (1999), and Jonker and Treur (1998) is adopted. This is a language in the family of languages to which also situation calculus (Reiter, 2001), event calculus (Kowalski and Sergot, 1986), and fluent calculus (Hölldobler and Tielscher, 1990) belong. In short, in TTL it is possible to express that in a given trace at a certain point in time the reasoning state has a certain (state) property. Moreover, it is possible to relate such state properties at different points in time. As an example, the following (global) property of a reasoning trace γ is considered, which expresses that all multiplication problems in two digits eventually will be solved.

GP1 (successfulness)
at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a multiplication problem for numbers x and y < 100 is present,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a solution z of this multiplication problem with $z = x*y$ is included.

The formalisation of this property in TTL is as follows.

$\forall t\ \forall x, y < 100$ state(γ, t, arithmetic) |== multiplication_problem(x, y)
$\Rightarrow\ \exists t' \geq t\ \exists z\quad z = x*y\ \ \&\ \$ state(γ, t', arithmetic) |== is_solution_for_multiplication_of(z, x, y)

Note that for simplicity no maximal allowed response time has been specified. If desired, this can be simply added by putting a condition t'≤r in the consequent with r the maximal response time. Similarly, other variants of overall properties can be specified, for example expressing that within the trace all multiplication problems will be solved without using any geometric or material representations. Moreover, instead of the arithmetical part of the reasoning state (arithmetic), again the specific internal or external arithmetical part (intarithmetic or extarithmetic) can be used, for example when expressing that only internal arithmetical representations are used. In the remaining of this paper, only the informal notation will be used for the properties.

5.1. Milestone Properties

Within the overall reasoning process a number of milestones can be defined, and properties can be identified that express whether the process from one milestone to another one has been performed properly. With respect to the geometrical reasoning, two intermediate milestones were defined: a reasoning state in which the problem has been represented in a geometric representation and it has been decomposed geometrically (after step 4 in the example trace), and a reasoning state in which a geometric representation with numbers in the areas occurs, i.e., in which the subproblems have been solved (after step 7 in the example trace). Accordingly, the following milestone properties have been formulated.

MP1
at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a multiplication problem for numbers x and y < 100 is present,
then a time point t'≥t exists such that in the reasoning state in γ at t' a geometric representation of a rectangle ABCD is included with points P on AB and Q on AD, with |AB| = x and |AD| = y
and this rectangle is partitioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC with |AP| = x_1, |PB| = x_2, |AQ| = y_1, and |QD| = y_2, where x_1, y_1 is the 10-part of x, resp. y, and x_2, y_2 is the digit part of x, resp. y.

Here, |AB| is the length of AB, and // is 'in parallel with'.

MP2
at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with points P on AB and Q on AD, with |AB| = x and |AD| = y,
and this rectangle is partitioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC with |AP| = x_1, |PB| = x_2, |AQ| = y_1, and |QD| = y_2, where x_1, y_1 is the 10-part of x, resp. y, and x_2, y_2 is the digit part of x, resp. y,
then a time point t'≥t exists such that in the reasoning state in γ at t' in each of these areas Aij a number zij is represented which equals $x_i * y_j$.

MP3
at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with |AB| = x and |AD| = y
and this rectangle is partitioned into four nonoverlapping rectangle areas A_{11}, A_{12}, A_{21}, A_{22},
and in each of these areas A_{ij} a number zij is represented which equals $x_i * y_j$, where x = x_1 + x_2, and y = y_1 + y_2,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a solution z with z = x*y of the multiplication problem (x, y) is included.

5.2. Local Properties

In this section a number of properties are identified that characterise the reasoning in a more local manner: each property characterises one reasoning step. For the sake of simplicity, for the example reasoning process persistence of representations in reasoning states over time is assumed, so that persistence does not need to be formulated within each of the properties.

LP1 (arithmetic-geometric)
at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a multiplication problem for numbers x and y < 100 is present,
then a time point t'≥t exists such that in the reasoning state in γ at t' a geometric representation of a rectangle ABCD with |AB| = x and |AD| = y is included.

This dynamic property expresses that in reasoning trace γ, if an arithmetically represented multiplication problem occurs, this eventually is translated into a geometric representation. The formalisation of this property in TTL is as follows.

∀t ∀x, y < 100 state(γ, t, arithmetic) |== multiplication_problem(x, y)
⇒ ∃t'≥t ∃A, B, C, D
state(γ, t', geometric) |== rectangle(A, B, C, D) & |AB| = x & |AD| = y

Further local properties are the following (not in any particular order).

LP2 (arithmetic-arithmetic)
at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a multiplication problem for numbers x and y < 100 is present,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a splitting of the numbers x and y in 'tens' and digits occurs, i.e., $x = x_1 + x_2$, $y = y_1 + y_2$ with x_1, y_1 multiples of 10 and x_2, y_2 <10.

LP3 (arithmetic-arithmetic)
at any point in time t
if the reasoning state in trace γ at t contains an arithmetic representation of a multiplication problem for (x, y), with x, y multiple of 10 or less than 10,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a solution z with $z = x*y$ for this multiplication problem for (x, y) is included.

LP4 (arithmetic-arithmetic)
at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of an addition problem for a finite list z_1 ,..., z_n of numbers of up to 4 digits is included,
then a time point t'≥t exists such that in the reasoning state in γ at t' a solution $z = \Sigma_{1\leq i \leq n} z_i$ of the addition problem is included.

LP5 (arithmetic-geometric)
at any point in time t
if in the reasoning state in trace γ at t an arithmetic representation of a splitting of the numbers x and y occurs, i.e., $x = x_1 + x_2$, $y = y_1 + y_2$,
then a time point t'≥t exists such that in the reasoning state in γ at t' a geometric representation of a rectangle ABCD with |AB| = x and |AD| = y is included with points P on AB and Q on AD such that $|AP| = x_1$, $|PB| = x_2$, $|AQ| = y_1$, and $|QD| = y_2$.

LP6 (geometric-geometric)
at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with points P on AB and Q on AD,
then a time point t'≥t exists such that in the reasoning state in γ at t' the rectangle ABCD is partitioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC.

LP7 (geometric-arithmetic)
at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD with |AB| = x and |AD| = y is included with points P on AB and Q on AD such that $|AP| = x_1$, $|PB| = x_2$, $|AQ| = y_1$, and $|QD| = y_2$,
and this rectangle is partioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC,
then a time point t'≥t exists such that in the reasoning state in γ at t' arithmetic representations of multiplication problems for (x_1, y_1), (x_1, y_2), (x_2, y_1), and (x_2, y_2) are included.

LP8 (geometric&arithmetic-geometric)
at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with points P on AB and Q on AD,
and this rectangle is partioned into four areas A_{11}, A_{12}, A_{21}, A_{22} by two lines PP'//AD and QQ'//AB with P' on CD and Q' on BC,
and arithmetic representations of solutions z_{11}, z_{12}, z_{21}, z_{22} for the multiplication problems for (|AP|, |AQ|), (|AP|, |QD|), (|PB|, |AQ|), and (|PB|, |QD|) are included.
then a time point t'≥t exists such that in the reasoning state in γ at t' within the geometric representation in each area A_{ij}, the number z_{ij} is represented.

LP9 (geometric-arithmetic)
at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included which is partioned into a number of areas A_1, …, A_n,
and within each of these areas A_i a number z_i is represented,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of an addition problem for z_1, …, z_n is included.

LP10 (geometric& arithmetic-arithmetic)
at any point in time t
if in the reasoning state in trace γ at t a geometric representation of a rectangle ABCD is included with |AB| = x and |AD| = y that is partitioned into a number of nonoverlapping areas A_1, …, A_n,
and within each of these areas A_i the number zi is represented,
and an arithmetic representation of a solution z of the addition problem for z_1, …, z_n is included,
then a time point t'≥t exists such that in the reasoning state in γ at t' an arithmetic representation of a solution z with z = x∗y of the multiplication problem (x, y) is included.

6. Dynamic Analysis

In this section it is described how the dynamic properties can be used for the analysis of existing reasoning processes.

6.1. Logical Relationships

A number of logical relationships have been established between the properties above. First of all, the three milestone properties together imply the global property:

MP1 & MP2 & MP3 ⇒ GP1 (0)

Next, each of these milestone properties is implied by a number of local properties:

LP1 & LP2 & LP5 & LP6 ⇒ MP1 (1)
LP3 & LP7 & LP8 ⇒ MP2 (2)
LP4 & LP9 & LP10 ⇒ MP3 (3)

These logical relationships can be depicted as an AND-tree, see Figure 4.

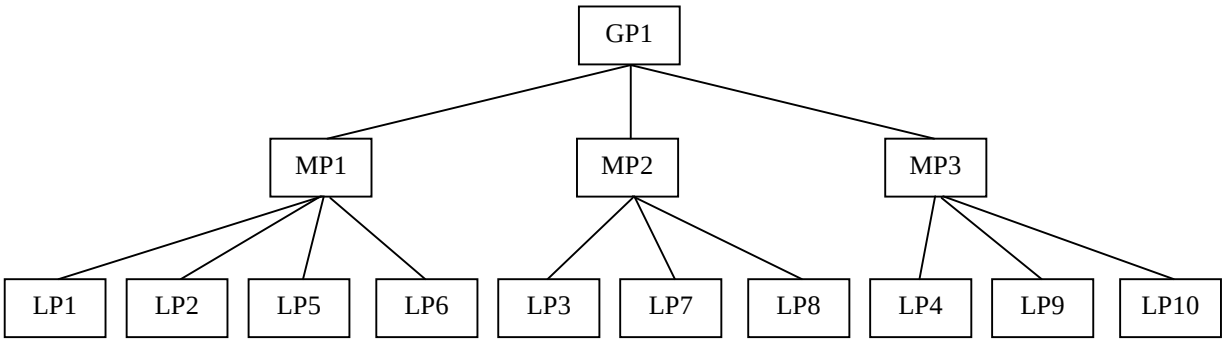


Figure 4. Logical relationships between dynamic properties

Identification of such logical relationships can be helpful in the analysis of errors within a given reasoning trace. For example, in case of a non-satisfactory reasoning trace it can first be checked whether **GP1** holds. If this global property does not hold, the three properties **MP1**, **MP2**, **MP3** can be checked. Given the logical relationship (0), at least one of

them will be found not to hold. This pinpoints the cause of the error in part of the process, say **MP3**. Next, (only) the local properties relating to **MP3** are checked, i.e, **LP4**, **LP7**, **LP10**, **LP11**. Again, due to (3) one of them will be found not to hold, which localises the cause of the error. Notice that this diagnostic process is economic in the sense that the whole subtrees under **MP1** and **MP2** are not examined as long as there is no reason for that.

6.2. Dynamic Analysis Method

Based on the idea of logical relationships between properties, a general analysis method for the dynamics of reasoning processes can be formulated. This analysis method comprises the following steps:

1) Identify the different dimensions or *components* of reasoning states.
2) Determine the different *types of transitions*.
3) Identify relevant *dynamic properties* for the reasoning
   a) for the process as a whole (global properties)
   b) for milestones within the process
   c) for reasoning steps (local properties)
4) Determine logical *relationships* between the different dynamic properties, in an AND-tree form; e.g.,
   a) local properties imply a milestone property, and
   b) milestone properties imply a global property.
5) For a given reasoning trace, *check* which of the dynamic properties hold and which do not hold. This can take the form of a diagnosis following the tree structure of the relationships between the dynamic properties. A software environment is available to support this checking process.

For the case at hand, more than 70 dynamic properties have been specified, varying from global properties for the overall reasoning process to more local properties. The idea is that some of these properties are of a general nature (i.e., they can be used to assess whether a trace qualifies as a proper reasoning trace), whereas the other properties are used to characterise the different types of possible traces (i.e., they are used to identify individual differences). A large number of automated checks have been performed, thereby checking dynamic properties as described in Section 5 against simulated traces as shown in Section 4.4, to reveal which properties hold for which traces. The results were in line with our expectations; for example, in the traces where all basic skills are present (e.g. the trace in Table 1), all properties of a general nature (such as the successfulness property GP1) turned out to hold. This validates the correctness of the simulation model, at least for the given traces. Likewise, in traces where the strategy profile described a preference for arithmetical representations, properties such as "*if possible, only arithmetic representations are used*" are satisfied.

In addition, note that the automated checker can also take empirical reasoning traces as input. Using this approach, in future research it will be checked which properties hold for empirical data, thereby supporting the comparison of human reasoning with simulated reasoning.

7. Discussion

Analysis of the cognitive capability to perform reasoning has been addressed from different areas and angles. Within Cognitive Science, the two dominant streams are the syntactic approach (based on inference rules applied to syntactic expressions, as common in logic), e.g., (Rips, 1994), and the semantic approach (based on construction of mental models); e.g., (Johnson-Laird, 1983). In experimental work for these approaches reasoning processes usually are studied by focussing on reasoning steps in isolation, by means of one-trial experiments. More extensive reasoning processes involving a number of steps that are tuned to each other require coherent controlled navigation. The current paper reports analysis and simulation of such a reasoning process.

The analysis method for the dynamics of reasoning processes used in this paper was adopted from (Jonker and Treur, 2002) and validated on the basis of reports from experiments with 8-9 year old children in classrooms in the Netherlands (Dekker et al., 1982). A similar report has been made by (Hutton, 1977). The current paper shows how an analysis of these dynamics can be made using traces consisting of sequences of reasoning states including control information over time to describe controlled reasoning processes. It is shown for the example reasoning pattern, how characterising dynamic properties can be identified. Furthermore, the agent modelling approach DESIRE has been used to implement a simulation model, and other software tools have been used to automatically check which dynamic properties hold for which simulated traces. In addition, these software tools can be used to check which properties hold for empirical data, thereby supporting the comparison of human reasoning with simulated reasoning. The variety of dynamic properties specified and the variety of traces simulated provides an overview for the individual differences between subjects that have been observed while solving multiplication problems. For example, using our formalisation those with an emphasis on external arithmetic representations are neatly distinguished from those who use external material representations where possible. In the analysis the notion of reasoning strategy was addressed, incorporating such differences in skill and preference. Due to the compositional structure of reasoning state it was not difficult to extend a reasoning state with a component for control information.

Note that the modelling approach used in this paper makes a clear distinction between generic and domain-specific aspects. This makes it relatively easy to plug in a different domain in multi-representational reasoning. For example, the domain can be modified to an example for children of 13 or 14 years to support algebra by geometric visualisations, e.g., the algebraic identity $(a + b)^2 = a^2 + 2ab + b^2$ interpreted as the area of a partitioned square of (a + b) x (a + b) in relation to areas of its parts: a square of a x a, a square of b x b, and two rectangles of a x b.

With respect to future work, further experiments will be conducted, in which also a focus is more explicitly on the control of the reasoning. For example, are subjects able to explain why at a point in time a translation to a geometric representation is made? Are think-aloud protocols involving control information a reliable source of further analysis? In addition, future work will explore the possibility to reuse the current simulation model in other cognitive domains.

Appendix A. Simulation Traces

This Appendix contains a number of simulation traces that were generated on the basis of the model described in Section 4.

Trace 1

Step	Information Derived
0	strategy profile: geo-ari-mat
0	available abstract skills: all skills
0	available mental skills: all skills
0	available physical skills: all skills
0	represent physically: all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))
2	plan([bs1, bs7, bs2, bs4, bs5, bs9, bs3, bs6, bs13, bs14])
3	is_represented_in_world(arithmetic, multiplication(23, 36))
4	mental_representation(geometric, entity(shape("[]"), parameters(23, 36)))
5	is_represented_in_world(geometric, rectangle('A', 'B', 'C', 'D', 23, 36))
6	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(36, 30, 6)))
6	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(23, 20, 3)))
7	is_represented_in_world(arithmetic, split(36, 30, 6))
7	is_represented_in_world(arithmetic, split(23, 20, 3))
8	mental_representation(geometric, entity(shape("-"), name('A', 'B'), parameters(20, 3)))
8	mental_representation(geometric, entity(shape("-"), name('A', 'D'), parameters(30, 6)))
9	is_represented_in_world(geometric, split('A', 'B', 20, 3))
9	is_represented_in_world(geometric, split('A', 'D', 30, 6))
10	mental_representation(geometric, entity(shape("[]"), name('A11'), parameters(20, 30)))
10	mental_representation(geometric, entity(shape("[]"), name('A12'), parameters(20, 6)))
10	mental_representation(geometric, entity(shape("[]"), name('A21'), parameters(3, 30)))
10	mental_representation(geometric, entity(shape("[]"), name('A22'), parameters(3, 6)))
11	is_represented_in_world(geometric, area('A11', 20, 30))
11	is_represented_in_world(geometric, area('A12', 20, 6))
11	is_represented_in_world(geometric, area('A21', 3, 30))
11	is_represented_in_world(geometric, area('A22', 3, 6))
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 6)))
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 30)))
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 6)))
12	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 30)))
13	is_represented_in_world(arithmetic, partial_multiplication('A11', 20, 30))
13	is_represented_in_world(arithmetic, partial_multiplication('A12', 20, 6))
13	is_represented_in_world(arithmetic, partial_multiplication('A21', 3, 30))
13	is_represented_in_world(arithmetic, partial_multiplication('A22', 3, 6))
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 6, 18)))
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 30, 90)))
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 6, 120)))
14	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 30, 600)))
15	is_represented_in_world(arithmetic, multiplication_solution(3, 6, 18))
15	is_represented_in_world(arithmetic, multiplication_solution(3, 30, 90))
15	is_represented_in_world(arithmetic, multiplication_solution(20, 6, 120))
15	is_represented_in_world(arithmetic, multiplication_solution(20, 30, 600))
16	mental_representation(geometric, entity(shape("[]"), name('A11'), area_with_number(600)))
16	mental_representation(geometric, entity(shape("[]"), name('A12'), area_with_number(120)))
16	mental_representation(geometric, entity(shape("[]"), name('A21'), area_with_number(90)))
16	mental_representation(geometric, entity(shape("[]"), name('A22'), area_with_number(18)))
17	is_represented_in_world(geometric, area_with_number('A11', 600))
17	is_represented_in_world(geometric, area_with_number('A12', 120))
17	is_represented_in_world(geometric, area_with_number('A21', 90))
17	is_represented_in_world(geometric, area_with_number('A22', 18))
18	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(600, 120, 90, 18)))
19	is_represented_in_world(arithmetic, addition(600, 120, 90, 18))
20	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(600, 120, 90, 18, 828)))
21	is_represented_in_world(arithmetic, addition_solution(600, 120, 90, 18, 828))
22	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))
23	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))

Trace 2

Step	Information Derived
0	strategy profile: ari-mat-geo
0	available abstract skills: all skills except bs10
0	available mental skills: all skills except bs10
0	available physical skills: all skills
0	represent physically: all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))
2	plan([bs24, bs25, bs9, bs26, bs13, bs14])
3	is_represented_in_world(arithmetic, multiplication(23, 36))
4	mental_representation(arithmetic, entity(shape("XY*"), parameters(23, 36)))
5	is_represented_in_world(arithmetic, symbolic_multiplication(23, 36))
6	mental_representation(arithmetic, entity(shape("X*Y"), parameters(2, 36)))
6	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 36)))
7	is_represented_in_world(arithmetic, partial_multiplication('A12', 3, 36))
7	is_represented_in_world(arithmetic, partial_multiplication('A11', 2, 36))
8	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(2, 36, 72)))
8	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 36, 108)))
9	is_represented_in_world(arithmetic, multiplication_solution(2, 36, 72))
9	is_represented_in_world(arithmetic, multiplication_solution(3, 36, 108))
10	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(108, 720, 0, 0)))
11	is_represented_in_world(arithmetic, addition(108, 720, 0, 0))
12	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(108, 720, 0, 0, 828)))
13	is_represented_in_world(arithmetic, addition_solution(108, 720, 0, 0, 828))
14	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))
15	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))

Trace 3

Step	Information Derived
0	strategy profile: mat-geo-ari
0	available abstract skills: all skills
0	available mental skills: all skills
0	available physical skills: all skills
0	represent physically: all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))
2	is_represented_in_world(arithmetic, multiplication(23, 36))
3	mental_representation(material, entity(shape("[]"), parameters(23, 36)))
3	mental_representation(material, entity(shape("[]"), size(big), number(0)))
3	mental_representation(material, entity(shape("[]"), size(medium_h), number(0)))
3	mental_representation(material, entity(shape("[]"), size(medium_v), number(0)))
3	mental_representation(material, entity(shape("[]"), size(small), number(0)))
4	is_represented_in_world(material, rectangle('A', 'B', 'C', 'D', 23, 36))
4	is_represented_in_world(material, block(big, 0))
4	is_represented_in_world(material, block(medium_h, 0))
4	is_represented_in_world(material, block(medium_v, 0))
4	is_represented_in_world(material, block(small, 0))
5	mental_representation(material, entity(shape("[]"), size(big), number(1)))
6	is_represented_in_world(material, block(big, 1))
7	mental_representation(material, entity(shape("[]"), size(big), number(2)))
8	is_represented_in_world(material, block(big, 2))
9	mental_representation(material, entity(shape("[]"), size(big), number(3)))
10	is_represented_in_world(material, block(big, 3))
11	mental_representation(material, entity(shape("[]"), size(big), number(4)))
12	is_represented_in_world(material, block(big, 4))
13	mental_representation(material, entity(shape("[]"), size(big), number(5)))
14	is_represented_in_world(material, block(big, 5))
15	mental_representation(material, entity(shape("[]"), size(big), number(6)))
16	is_represented_in_world(material, block(big, 6))
17	mental_representation(material, entity(shape("[]"), size(big), total(6)))
18	is_represented_in_world(material, total_of_blocks(big, 6))
19	mental_representation(material, entity(shape("[]"), size(medium_h), number(1)))
20	is_represented_in_world(material, block(medium_h, 1))
21	mental_representation(material, entity(shape("[]"), size(medium_h), number(2)))
22	is_represented_in_world(material, block(medium_h, 2))
23	mental_representation(material, entity(shape("[]"), size(medium_h), number(3)))
24	is_represented_in_world(material, block(medium_h, 3))
25	mental_representation(material, entity(shape("[]"), size(medium_h), number(4)))
26	is_represented_in_world(material, block(medium_h, 4))
27	mental_representation(material, entity(shape("[]"), size(medium_h), number(5)))
28	is_represented_in_world(material, block(medium_h, 5))
29	mental_representation(material, entity(shape("[]"), size(medium_h), number(6)))

30	is_represented_in_world(material, block(medium_h, 6))
31	mental_representation(material, entity(shape("[]"), size(medium_h), number(7)))
32	is_represented_in_world(material, block(medium_h, 7))
33	mental_representation(material, entity(shape("[]"), size(medium_h), number(8)))
34	is_represented_in_world(material, block(medium_h, 8))
35	mental_representation(material, entity(shape("[]"), size(medium_h), number(9)))
36	is_represented_in_world(material, block(medium_h, 9))
37	mental_representation(material, entity(shape("[]"), size(medium_h), number(10)))
38	is_represented_in_world(material, block(medium_h, 10))
39	mental_representation(material, entity(shape("[]"), size(medium_h), number(11)))
40	is_represented_in_world(material, block(medium_h, 11))
41	mental_representation(material, entity(shape("[]"), size(medium_h), number(12)))
42	is_represented_in_world(material, block(medium_h, 12))
43	mental_representation(material, entity(shape("[]"), size(medium_h), total(12)))
44	is_represented_in_world(material, total_of_blocks(medium_h, 12))
45	mental_representation(material, entity(shape("[]"), size(medium_v), number(1)))
46	is_represented_in_world(material, block(medium_v, 1))
47	mental_representation(material, entity(shape("[]"), size(medium_v), number(2)))
48	is_represented_in_world(material, block(medium_v, 2))
49	mental_representation(material, entity(shape("[]"), size(medium_v), number(3)))
50	is_represented_in_world(material, block(medium_v, 3))
51	mental_representation(material, entity(shape("[]"), size(medium_v), number(4)))
52	is_represented_in_world(material, block(medium_v, 4))
53	mental_representation(material, entity(shape("[]"), size(medium_v), number(5)))
54	is_represented_in_world(material, block(medium_v, 5))
55	mental_representation(material, entity(shape("[]"), size(medium_v), number(6)))
56	is_represented_in_world(material, block(medium_v, 6))
57	mental_representation(material, entity(shape("[]"), size(medium_v), number(7)))
58	is_represented_in_world(material, block(medium_v, 7))
59	mental_representation(material, entity(shape("[]"), size(medium_v), number(8)))
60	is_represented_in_world(material, block(medium_v, 8))
61	mental_representation(material, entity(shape("[]"), size(medium_v), number(9)))
62	is_represented_in_world(material, block(medium_v, 9))
63	mental_representation(material, entity(shape("[]"), size(medium_v), total(9)))
64	is_represented_in_world(material, total_of_blocks(medium_v, 9))
65	mental_representation(material, entity(shape("[]"), size(small), number(1)))
66	is_represented_in_world(material, block(small, 1))
67	mental_representation(material, entity(shape("[]"), size(small), number(2)))
68	is_represented_in_world(material, block(small, 2))
69	mental_representation(material, entity(shape("[]"), size(small), number(3)))
70	is_represented_in_world(material, block(small, 3))
71	mental_representation(material, entity(shape("[]"), size(small), number(4)))
72	is_represented_in_world(material, block(small, 4))
73	mental_representation(material, entity(shape("[]"), size(small), number(5)))
74	is_represented_in_world(material, block(small, 5))
75	mental_representation(material, entity(shape("[]"), size(small), number(6)))
76	is_represented_in_world(material, block(small, 6))
77	mental_representation(material, entity(shape("[]"), size(small), number(7)))
78	is_represented_in_world(material, block(small, 7))
79	mental_representation(material, entity(shape("[]"), size(small), number(8)))
80	is_represented_in_world(material, block(small, 8))
81	mental_representation(material, entity(shape("[]"), size(small), number(9)))
82	is_represented_in_world(material, block(small, 9))
83	mental_representation(material, entity(shape("[]"), size(small), number(10)))
84	is_represented_in_world(material, block(small, 10))
85	mental_representation(material, entity(shape("[]"), size(small), number(11)))
86	is_represented_in_world(material, block(small, 11))
87	mental_representation(material, entity(shape("[]"), size(small), number(12)))
88	is_represented_in_world(material, block(small, 12))
89	mental_representation(material, entity(shape("[]"), size(small), number(13)))
90	is_represented_in_world(material, block(small, 13))
91	mental_representation(material, entity(shape("[]"), size(small), number(14)))
92	is_represented_in_world(material, block(small, 14))
93	mental_representation(material, entity(shape("[]"), size(small), number(15)))
94	is_represented_in_world(material, block(small, 15))
95	mental_representation(material, entity(shape("[]"), size(small), number(16)))
96	is_represented_in_world(material, block(small, 16))
97	mental_representation(material, entity(shape("[]"), size(small), number(17)))
98	is_represented_in_world(material, block(small, 17))
99	mental_representation(material, entity(shape("[]"), size(small), number(18)))
100	is_represented_in_world(material, block(small, 18))
101	mental_representation(material, entity(shape("[]"), size(small), total(18)))
102	is_represented_in_world(material, total_of_blocks(small, 18))
103	mental_representation(arithmetic, entity(shape("X*Y"), parameters(18, 1)))
104	is_represented_in_world(arithmetic, partial_multiplication('A22', 18, 1))
105	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(18, 1, 18)))
106	is_represented_in_world(arithmetic, multiplication_solution(18, 1, 18))
107	mental_representation(material, entity(shape("[]"), size(small), area_with_number(18)))
108	is_represented_in_world(material, area_with_number(small, 18))
109	mental_representation(arithmetic, entity(shape("X*Y"), parameters(9, 10)))
110	is_represented_in_world(arithmetic, partial_multiplication('A21', 9, 10))
111	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(9, 10, 90)))

112	is_represented_in_world(arithmetic, multiplication_solution(9, 10, 90))
113	mental_representation(material, entity(shape("[]"), size(medium_v), area_with_number(90)))
114	is_represented_in_world(material, area_with_number(medium_v, 90))
115	mental_representation(arithmetic, entity(shape("X*Y"), parameters(12, 10)))
116	is_represented_in_world(arithmetic, partial_multiplication('A12', 12, 10))
117	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(12, 10, 120)))
118	is_represented_in_world(arithmetic, multiplication_solution(12, 10, 120))
119	mental_representation(material, entity(shape("[]"), size(medium_h), area_with_number(120)))
120	is_represented_in_world(material, area_with_number(medium_h, 120))
121	mental_representation(arithmetic, entity(shape("X*Y"), parameters(6, 100)))
122	is_represented_in_world(arithmetic, partial_multiplication('A22', 6, 100))
123	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(6, 100, 600)))
124	is_represented_in_world(arithmetic, multiplication_solution(6, 100, 600))
125	mental_representation(material, entity(shape("[]"), size(big), area_with_number(600)))
126	is_represented_in_world(material, area_with_number(big, 600))
127	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(600, 120, 90, 18)))
128	is_represented_in_world(arithmetic, addition(600, 120, 90, 18))
129	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(600, 120, 90, 18, 828)))
130	is_represented_in_world(arithmetic, addition_solution(600, 120, 90, 18, 828))
131	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))
132	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))

Trace 4

Step	Information Derived	
0	strategy profile:	ari-geo-mat
0	available abstract skills:	all skills except bs10
0	available mental skills:	all skills except bs10 and bs25
0	available physical skills:	all skills
0	represent physically:	all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))	
2	plan([bs24, bs25, bs9, bs26, bs13, bs14])	
3	is_represented_in_world(arithmetic, multiplication(23, 36))	
4	mental_representation(arithmetic, entity(shape("XY*"), parameters(23, 36)))	
5	is_represented_in_world(arithmetic, symbolic_multiplication(23, 36))	
	/* mental part of bs25 fails -> create new plan */	
6	plan([bs7, bs1, bs2, bs4, bs5, bs9, bs3, bs6, bs13, bs14])	
7	is_represented_in_world(arithmetic, multiplication(23, 36))	
8	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(36, 30, 6)))	
8	mental_representation(arithmetic, entity(shape("X=X1+X2"), parameters(23, 20, 3)))	
9	is_represented_in_world(arithmetic, split(36, 30, 6))	
9	is_represented_in_world(arithmetic, split(23, 20, 3))	
10	mental_representation(geometric, entity(shape("[]"), parameters(23, 36)))	
11	is_represented_in_world(geometric, rectangle('A', 'B', 'C', 'D', 23, 36))	
12	mental_representation(geometric, entity(shape("-"), name('A', 'B'), parameters(20, 3)))	
12	mental_representation(geometric, entity(shape("-"), name('A', 'D'), parameters(30, 6)))	
13	is_represented_in_world(geometric, split('A', 'B', 20, 3))	
13	is_represented_in_world(geometric, split('A', 'D', 30, 6))	
14	mental_representation(geometric, entity(shape("[]"), name('A11'), parameters(20, 30)))	
14	mental_representation(geometric, entity(shape("[]"), name('A12'), parameters(20, 6)))	
14	mental_representation(geometric, entity(shape("[]"), name('A21'), parameters(3, 30)))	
14	mental_representation(geometric, entity(shape("[]"), name('A22'), parameters(3, 6)))	
15	is_represented_in_world(geometric, area('A11', 20, 30))	
15	is_represented_in_world(geometric, area('A12', 20, 6))	
15	is_represented_in_world(geometric, area('A21', 3, 30))	
15	is_represented_in_world(geometric, area('A22', 3, 6))	
16	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 6)))	
16	mental_representation(arithmetic, entity(shape("X*Y"), parameters(3, 30)))	
16	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 6)))	
16	mental_representation(arithmetic, entity(shape("X*Y"), parameters(20, 30)))	
17	is_represented_in_world(arithmetic, partial_multiplication('A11', 20, 30))	
17	is_represented_in_world(arithmetic, partial_multiplication('A12', 20, 6))	
17	is_represented_in_world(arithmetic, partial_multiplication('A21', 3, 30))	
17	is_represented_in_world(arithmetic, partial_multiplication('A22', 3, 6))	
18	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 6, 18)))	
18	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(3, 30, 90)))	
18	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 6, 120)))	
18	mental_representation(arithmetic, entity(shape("X*Y=Z"), parameters(20, 30, 600)))	
19	is_represented_in_world(arithmetic, multiplication_solution(3, 6, 18))	
19	is_represented_in_world(arithmetic, multiplication_solution(3, 30, 90))	
19	is_represented_in_world(arithmetic, multiplication_solution(20, 6, 120))	
19	is_represented_in_world(arithmetic, multiplication_solution(20, 30, 600))	

20	mental_representation(geometric, entity(shape("[]"), name('A11'), area_with_number(600)))
20	mental_representation(geometric, entity(shape("[]"), name('A12'), area_with_number(120)))
20	mental_representation(geometric, entity(shape("[]"), name('A21'), area_with_number(90)))
20	mental_representation(geometric, entity(shape("[]"), name('A22'), area_with_number(18)))
21	is_represented_in_world(geometric, area_with_number('A11', 600))
21	is_represented_in_world(geometric, area_with_number('A12', 120))
21	is_represented_in_world(geometric, area_with_number('A21', 90))
21	is_represented_in_world(geometric, area_with_number('A22', 18))
22	mental_representation(arithmetic, entity(shape("V+W+X+Y"), parameters(600, 120, 90, 18)))
23	is_represented_in_world(arithmetic, addition(600, 120, 90, 18))
24	mental_representation(arithmetic, entity(shape("V+W+X+Y=Z"), parameters(600, 120, 90, 18, 828)))
25	is_represented_in_world(arithmetic, addition_solution(600, 120, 90, 18, 828))
26	mental_representation(arithmetic, entity(shape("XX*YY=ZZ"), parameters(23, 36, 828)))
27	is_represented_in_world(arithmetic, multiplication_solution(23, 36, 828))

Trace 5

Step	Information Derived	
0	strategy profile:	ari-geo-mat
0	available abstract skills:	all skills except bs7, bs15, bs24
0	available mental skills:	all skills
0	available physical skills:	all skills
0	represent physically:	all steps
1	mental_representation(arithmetic, entity(shape("X*Y"), parameters(23, 36)))	
	/* no further derivations: agent fails to make a plan */	

References

Booker, G., Bond, D., Briggs, J. & Davey, G. (1997). Teaching Primary Mathematics, Melbourne: Longman.

Brazier, F. M. T., Jonker, C.M., and Treur, J., (2002). Principles of Component-Based Design of Intelligent Agents. Data and Knowledge Engineering, vol. 41, pp. 1-28.

Bruner, J.S. (1968). *Toward a Theory of Instruction*. Norton & Company, Inc. New York.

Dekker, A., Heege, H. ter, and Treffers, A. (1982). *Cijferend vermenigvuldigen en delen volgens Wiskobas*. Universiteit Utrecht, Freudenthal Institute.

English, L. & Halford, G. (1995) Mathematics Education: Models and Processes. Mahwah, NJ: Lawrence Erlbaum

Hegarty, M. (2002). Mental Visualizations and External Visualizations. In: W.D. Gray and C.D. Schunn (eds.), Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci'02. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2002, p. 40.

Herlea, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E. (1999). Specification of Behavioural Requirements within Compositional Multi-Agent System Design. In: F.J. Garijo, M. Boman (eds.), *Multi-Agent System Engineering, Proc. of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99*. Lecture Notes in AI, vol. 1647, Springer Verlag, 1999, pp. 8-27.

Hölldobler, S., and Thielscher, M. (1990). A new deductive approach to planning. *New Generation Computing*, 8:225-244, 1990.

Hutton, J. (1977). Memoirs of a Maths Teacher 5: Logical Reasoning. In: *Mathematics Teaching*, vol. 81, pp. 8-12.

Johnson-Laird, P.N. (1983). *Mental Models*. Cambridge: Cambridge University Press.

Jonker, C.M., and Treur, J. (1998). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), *Proceedings of the International Workshop on Compositionality, COMPOS'97*. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380.

Extended version in: *International Journal of Cooperative Information Systems.* To appear, 2002.

Koedinger, K.R., and Terao, A. (2002). A Cognitive Task Analysis of Using Pictures To Support Pre-Algebraic Reasoning. In: W.D. Gray and C.D. Schunn (eds.), Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci'02. Mahwah, NJ: Lawrence Erlbaum Associates, 2002, pp. 542-547.

Kowalski, R., and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4:67-95, 1986.

Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

Rips, L.J. (1994). *The Psychology of Proof: Deductive reasoning in human thinking*. MIT Pres, Cambridge, Mass.

Yang, Y., and Johnson-Laird, P.N. (1999). A study of complex reasoning: The case GRE 'logical' problems. In M. A. Gernsbacher & S. J. Derry (Eds.) *Proceedings of the Twenty First Annual Conference of the Cognitive Science Society*, 767-771.

PART II REASONING

CHAPTER 7

Analysis of Design Process Dynamics

This chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2004). Analysis of Design Process Dynamics. In: Lopez de Mantaras, R. and Saitta, L. (eds.), *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'04*, IOS Press, pp. 293-297.

Analysis of Design Process Dynamics

Tibor Bosse[1], Catholijn M. Jonker[1], and Jan Treur[1,2]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, jonker, treur}@cs.vu.nl
http://www.cs.vu.nl/~{tbosse, jonker, treur}
[2] Universiteit Utrecht, Department of Philosophy,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Abstract. To enable the development of automated support for design, a challenge is to model and analyse dynamics of design processes in a formal manner. This paper contributes a declarative, logical approach for specification of dynamic properties of design processes, supported by a formal temporal language. This language is used to specify dynamic properties of a design process as a whole, or of parts thereof. At the most detailed level, in an executable sublanguage also simulation models are specified in a declarative, logical manner, which allows to use these specifications in logical analysis as well. The approach is illustrated by an example component-based agent-system design process.

1. Introduction

Providing automated support to manage the dynamics of a design process is in most cases not trivial. For example, in [6] some of the requirements put forward are that (1) a complete design process representation is needed, (2) with sufficient detail to allow for direct execution. Also by [1], [5] it is put forward that supporting the management of the dynamics of a design process is an important challenge to be addressed. This indeed is the aim of the current paper. The type of design considered is the design of component-based (e.g., software) systems for dynamic applications. In such application areas often components can be (re)used for which the properties are known. By composing a number of such components in a component-based design, the required overall dynamics is obtained. As holds for many design processes, designing component-based systems can be a rather complex and dynamic process, for which a number of tasks play a role, for example in this specific case:

1) maintaining of specifications of properties of (reusable) components
2) maintaining of requirements on the overall system to be designed (usually in close contact with a stakeholder)
3) refinement and revision of requirements
4) determination of reusable components based on their properties, to find a system that satisfies the requirements
5) checking whether a system (a design object description) satisfies the requirements
6) revision of a design object description that does not satisfy the requirements

Most of these tasks essentially involve the dynamics of design as a process. The analysis of this *design process dynamics* is the subject of the current paper.

During a design process, two important concepts play a role: a design problem statement and a solution specification. A *design problem statement* consists of:

- a set of requirements in the form of dynamic properties on the overall system behaviour that have to be fulfilled
- a partial description of (prescribed) system architecture that has to be incorporated
- a partial description of (prescribed) dynamic properties of elements of the system that have to be incorporated; e.g., for components, for transfers, for parts, for interactions between parts.

A *solution specification* for a design problem is a specification of a design object (both structure and behaviour) that fulfils the imposed requirements on overall behaviour, and includes the given (prescribed) descriptions of structure and behaviour. Here 'fulfilling' the overall behaviour requirements means that they are implied by the dynamic properties for components, transfers and interactions between parts within the specification.

In this paper, in Section 2 a formalisation of design process dynamics will be discussed in terms of design states, design transitions and design traces. Section 3 addresses some dynamic properties of design processes. Section 4 gives an overview of an example design process. In Section 5, a relevant requirement will be given for the example system to be designed. It will be shown how this global requirement for the overall system can be refined to local requirements for parts of the system. Section 6 will describe a simulation model of the example design process, and shows an example simulation trace. In Section 7 the example design process is analysed in terms of dynamic properties. Finally, Section 8 is a conclusion.

2. Design Process Dynamics

To analyse dynamics of a design process, a formalisation is needed of such dynamics. Such a formalisation is introduced in this section, inspired by [4]. It is based on the notion of design process state and design trace.

The state of a design process at a certain time point can be described as a combined state consisting of two states, S = <S1, S2> with:

- S1 requirements manipulation state (RM-state),

  including the current requirements set
- S2 design object description manipulation state (DM-state),

  including the current design object description state

A particular design process shows a sequence of transitions from one state S to another (next) state S'. Design *traces* are time-indexed sequences of design states, where each subsequent pair of states is a design transition. To describe such sequences a fixed *time frame* T is assumed which is linearly ordered. A *trace* γ over a state ontology Ont (including ontology for design objects and requirements) and time frame T is a mapping $\gamma : T \rightarrow STATES(Ont)$, i.e., a sequence of states γ_t ($t \in T$) in $STATES(Ont)$. The set of all traces over state ontology Ont is denoted by $TRACES(Ont)$. Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering.

3. Dynamic Properties of Design

To formally specify dynamic properties that express characteristics of dynamic processes (such as design) from a temporal perspective an expressive language is needed. To this end the *Temporal Trace Language* TTL is used as a tool; cf. [7], which is briefly defined as follows. The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, a certain state during a design process at time point t is denoted by state(γ, t), which as a TTL-expression refers to γ_t. These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t) |= p denotes that state property p holds in trace γ at time t. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts T for time points, Traces for traces and F for state formulae, using quantifiers over time and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$.

To be able to perform some automated experiments with design processes, a simpler language has been used. This language (the *leads to* language) enables to model direct temporal dependencies between two state properties in successive states, as occur in specifications of a simulation model (for example, if in the current state, state property p holds, then in the next state, state property q holds). This language enables the automatic generation of simulated traces. The executable format is defined as follows. Let α and β be state properties of the form 'conjunction of ground atoms or negations of ground atoms. In the leads to language the notation $\alpha \rightarrow\!\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a formal definition of the leads to language in terms (as a sublanguage) of the language TTL, see [8].

Two different types of dynamic properties can be distinguished: Local Properties and Global Properties. Local properties only concern the smallest steps (taken into account in the conceptualisation of the process) in the process under analysis; for example:

At every point in time,
if a requirement **r** is imposed on the object to be designed,
and this requirement can be refined to sub-requirement **q**
then at the next point in time, sub-requirement **q** will be imposed on the object to be designed

In contrast, a global property is a property that concerns the overall process (taken into account) in the process under analysis, for example:

Eventually there is a committed requirement set **R** and
a design object description **D** such that, for each requirement **r** in **R**,
the design object description **D** satisfies requirement **r**

More complex Local and Global dynamic properties for design processes will be introduced in Sections 6 and 7, respectively.

4. An Example Design Process

To address in more detail the analysis of design process dynamics, an example design process was taken. The analysis approach is described and evaluated for this example design process. The example design process concerns the design of a cooperative

information gathering agent system (see Section 4.2). The design approach is by requirements refinement (see Section 4.1).

4.1. Design by Requirements Refinement

A design process of a complex system (e.g., a software system) usually starts by specifying requirements for the overall system behaviour. They express the dynamic properties that should 'emerge' if appropriate components are designed and combined in a proper manner. Given these requirements on overall system behaviour, the system is designed in such a manner that the requirements are fulfilled.

Between dynamic properties at different levels of aggregation within a complex system (to be) designed, certain interlevel relationships can be identified; overall behaviour of the design object can be related to dynamic properties of parts of the design object and properties of interaction between these parts via the following pattern:

dynamic properties for the parts & dynamic properties for interaction between parts ⇒
dynamic properties for the design object.

The process to identify new, refined requirements for behaviour of parts of the system and their interaction is called *requirements refinement*. Subsequently, the required dynamic properties of parts can be refined to dynamic properties of certain components and transfers, making use of:

dynamic properties for components & dynamic properties for transfer between components ⇒
dynamic properties for a part.

4.2. An Example Design Problem

As a case study, the process of designing a multi-agent system for cooperative information gathering [7] will be analysed in more detail. To get the idea, assume the system to be designed has to consist of three agents: A, B and C. The resulting behaviour of the system must be as follows: agent A and B are able to do some investigations and make up a report on some topic, and communicate that to the third agent C. Both A and B have access to useful sources of information, but this differs for the two agents. By co-operation they can benefit from the exchange of information that is only accessible to the other agent. If both types of information are combined, conclusions can be drawn that would not have been achievable for each of the agents separately. To make the example more precise: the example agent model is composed of three components: two information gathering agents A and B, agent C, and environment component E representing the external world, see Figure 1.

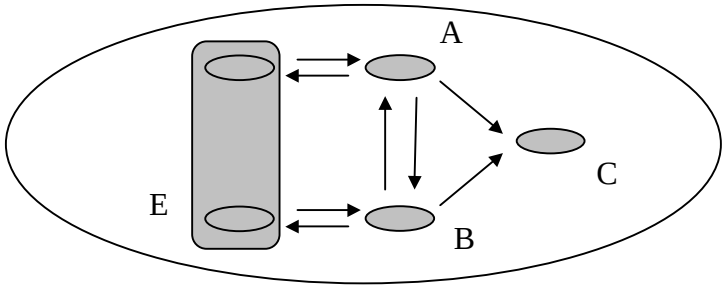


Figure 1. The example Agent System

Each of the agents is able to acquire partial information from an external source by initiated observations. For reasons of presentation, this by itself quite common situation for co-operative information agents is materialised in the following more concrete form. The world situation consists of an object that has to be classified. One agent can observe only the bottom view of the object, the other agent only the side view. By exchanging and combining observation information they are able to classify the object. For example, if an agent knows that the views are a circle and a square, it is concluded that the object is a cylinder.

In most multi-agent systems it is common that each agent has its own characteristics or attitudes. In the current system to be designed, the agents used as components in the design can differ, for instance, in their attitudes towards observation and communication: an agent may or may not be *pro-active*, in the sense that it takes the initiative with respect to one or more of:

- performing observations
- communicate its own observation results to the other agent
- ask the other agent for its observation results
- draw conclusions about the classification of the object

Moreover, an agent may be *reactive* to the other agent in the sense that it responds to a request for observation information:

- by communicating its observation result as soon as they are available
- by starting to observe for the other agent

The successfulness of the system to be designed will depend on the combination of attitudes of the agents. For example, if both agents are pro-active and reactive in all respects, then they can easily come to a conclusion. However, it is also possible that one of the agents is only reactive, and still the other agent comes to a conclusion. So, successfulness can be achieved in many ways and depends on subtle interactions between pro-activeness and reactiveness attitudes of both agents.

5. Requirements of the Example

In this section the example agent system to be designed as discussed in the previous section is further elaborated in terms of relevant requirements. Therefore, it is necessary to define the design problem statement, consisting of the requirements on the overall agent system behaviour. To simplify the example, it is assumed that just one main requirement is imposed on the current agent system, namely is whether or not a result will be generated. This requirement is called DODGP (Design Object Description Global Property):

DODGP Successfulness
For any trace of the system, there exists a point in time such that in this trace at that point in time agent C will receive a correct conclusion, either from A or from B (or from both).

In virtue of which combination of dynamic properties of the agents can success be achieved? In other words, which dynamic properties for the agents imply the property successfulness? Such a requirements refinement process can be managed more effectively if the overall requirements are not directly related to agent behaviour requirements, but one or more intermediate levels are created, as explained in Section 4.1. The idea is that for the agent system to be successful it is needed that

- both information sources within the environment E are addressed,
- if they are addressed, they provide the relevant information, and
- if the relevant information is provided by the information sources, a conclusion is drawn.

This first requirements refinement provides the dynamic properties DODGP1, DODGP2, DODGP3:

DODGP1 Information request effectiveness
At some points in time A and B will start information acquisition to E.

DODGP2 Information source effectiveness
If at some points in time A and B start information acquisition to E,
then E will generate all the correct relevant information for both.

DODGP3 Concluding effectiveness
If at some points in time E generates all the correct relevant information, then C will receive a correct conclusion.

These properties are logically related to DODGP (see also Table 1) by the implication: DODGP1 & DODGP2 & DODGP3 $\Rightarrow$ DODGP.

A next step in the requirements refinement process is to relate each of the dynamic properties DODGP1, DODGP2 and DODGP3 to agent behaviour properties. The complete refinement of these properties is elaborated in [2]. Due to space limitations, in this paper we only present a table with logical relationships between dynamic properties, without showing the exact definitions of all of the properties.

DODGP1 $\wedge$ DODGP2 $\wedge$ DODGP3	$\Rightarrow$	DODGP
B1 $\vee$ B2 $\vee$ B3	$\Rightarrow$	DODGP1
DODI1(A) $\wedge$ DODI1(B)	$\Rightarrow$	DODGP2
DODI2(X,Y) $\wedge$ DODI3(X,Y,C)	$\Rightarrow$	DODGP3
DODBP1(A) $\wedge$ DODBP1(B)	$\Rightarrow$	B1
DODBP1(X) $\wedge$ DODBP2(X) $\wedge$ DODBP4(Y) $\wedge$ DODTP(X,Y)	$\Rightarrow$	B2
DODBP2(A) $\wedge$ DODBP4(A) $\wedge$ DODBP2(B) $\wedge$ DODBP4(B) $\wedge$ DODTP(A,B) $\wedge$ DODTP(B,A)	$\Rightarrow$	B3
DODEP(A) $\wedge$ DODTP(A,E)	$\Rightarrow$	DODI1(A)
DODEP(B) $\wedge$ DODTP(B,E)	$\Rightarrow$	DODI1(B)
B4 $\vee$ B5	$\Rightarrow$	DODI2(X,Y)
DODBP3(X) $\wedge$ DODTP(Y,X) $\wedge$ DODTP(E,X) $\wedge$ DODTP(X,C)	$\Rightarrow$	DODI3(X,Y,C)
DODBP6(Y) $\wedge$ DODTP(E,Y)	$\Rightarrow$	B4
DODBP2(X) $\wedge$ DODBP5(Y) $\wedge$ DODTP(X,Y) $\wedge$ DODTP(E,Y)	$\Rightarrow$	B5

Table 1. Overview of all possible requirement refinements

In Table 1, in the form of logical implications an overview is shown of all possible refinements as discussed. Here X and Y are variables over the set {agent A, agent B}, where $X \neq Y$. Note that the different alternatives (*branches*) are indicated by the names B1 to B5. Moreover, to be able to distinguish the properties concerning the system to be designed (presented in this section) from the properties concerning the design process itself (presented in Section 7), the names of the former have been slightly modified with respect to [2]. To be specific, to the name of each property, the prefix "DOD" has been added.

6. A Simulation Model

Making use of the formal approach described in Section 3, the dynamics of the example design process have been simulated by means of *local properties*. Two types of local properties are distinguished: those that model the dynamics of requirements states, and those that model the dynamics of the Design Object Description states. Due to space limitations, only a subset of the Local Properties used for the simulation are shown.

The process concerning *requirements* takes into account whether or not the stakeholder asserts that certain requirements are undesirable.

LP4 Requirement Refinement
Local property LP4 expresses that, if currently a requirement p exists that can be refined to a subrequirement q, and it has not been refined yet, then this should be done by refining via the best branch b (e.g. the one with the lowest costs). Formalisation:
is_a_current_requirement(p) ∧ is_a_subrequirement_of_via(q,p,b) ∧
not(requirement_refined(p)) ∧ best_branch_for(b,p) ∧ not(undesirable_branch(b))
→→ is_a_current_requirement(q) ∧ requirement_refined(p) ∧
requirement_refined_via(p,b)

The process concerning *Design Object Descriptions* determines design object descriptions for sets of requirements given as input. Within this process it is taken into account whether or not the stakeholder asserts that certain components are undesirable as part of a design object.

LP6 DOD Generation
This property expresses that each local requirement l should be satisfied by adding the best component c for that requirement to the current DOD dod(x). Formalisation:
'DOD_counter'(x) ∧ is_a_current_requirement(l) ∧ best_component_for(c,l) ∧
not(undesirable_component(c)) →→ current_DOD(dod(x)) ∧ part_of_DOD(c,dod(x))

LP8 Local Requirement Satisfaction Determination
This property determines when a local requirement l is satisfied by a DOD. This is the case when the current DOD contains a component c for which this requirement holds. Formalisation:
current_DOD(d) ∧ part_of_DOD(c,d) ∧ holds_for(l,c) ∧ is_a_current_requirement(l)
→→ local_requirement_satisfied(l)

Using the simulation model, a number of experiments have been performed. In such experiments, different types of revision might be needed with an increasing impact on the design process:

- revision of the *design object description* for given requirements based on the stakeholders judgement that a component used in the DOD is undesirable.
- revision of the refined *requirements* based on the stakeholder's judgement that one of these requirements is undesirable.
- revision of a whole *branch* based on the calculation that the costs of the design object description found are higher than expected.

An example trace of a design process in which the last type of revision is needed is depicted in Figure 2. Time is on the horizontal axis, the derived state properties are on the vertical axis. In this simulation, for all local properties the values (0,0,1,1) have been chosen for the timing parameters e, f, g, and h. Due to space limitations, only a subset of the derived atoms is shown, but the overall dynamics of the process are clear:

When the process starts, first the initial requirement dodgp is identified. After this, this requirement is refined into sub-requirements dodgp1, dodgp2 and dodgp3 (based on the logical relationships of Table 1). This process continues until the most elementary

requirements (i.e. those that have no subrequirements) have been reached. Then a new design object description (called dod(1)) is created which consists of a number of components that satisfy all local requirements. Based on the costs of these components, the system calculates the total costs for each branch (i.e., for each collection of subrequirements, see Table 1). In case this number is higher than the predicted costs for that branch, the branch is marked as undesirable. This turns out to be the case at, for example, time point 17. Here, the refinement of requirement dodi2(x,y) via branch b4 turns out to be too expensive. As a result, the system starts backtracking in the table of logical relationships in order to select another branch. In total, three branches are revised in this trace (namely b4, b1 and b2, respectively). Finally, the system succeeds in finding a satisfactory DOD. This resulting DOD is then evaluated and its total costs are calculated.

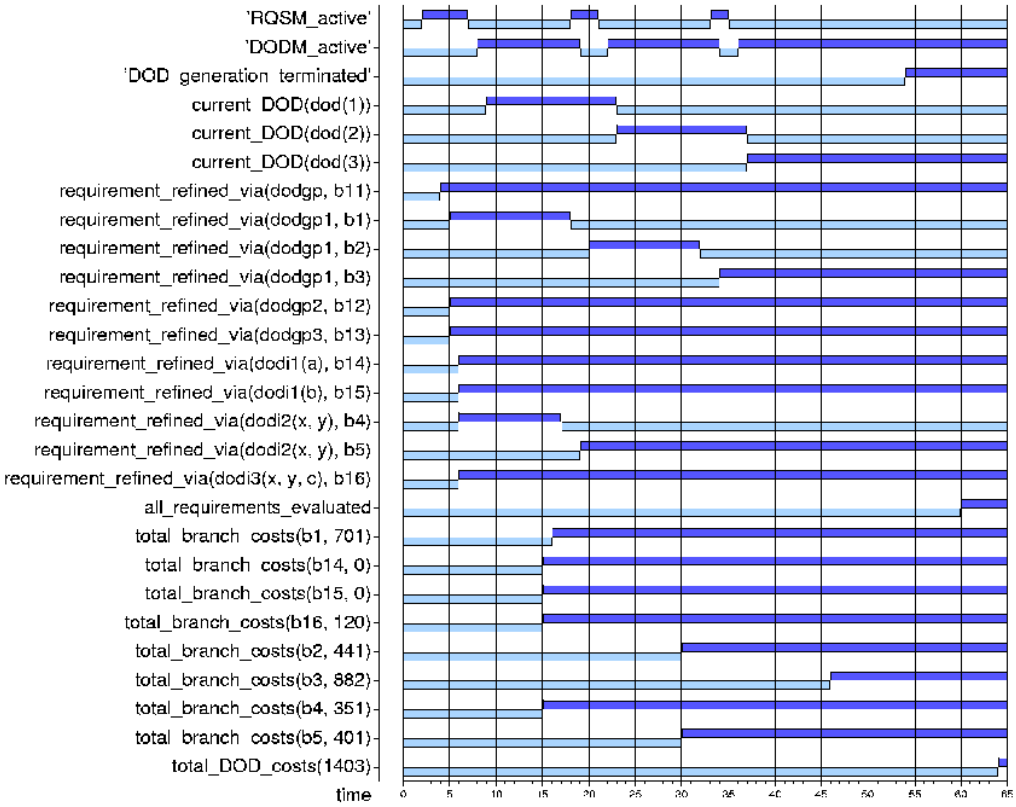


Figure 2. Simulation trace

7. Global Dynamic Properties

For design processes like the one described above, a number of global dynamic properties can be identified. For example:

- During (or after termination of) the design process, the design process objectives are fulfilled
- After termination of the design process the final design object description satisfies the requirements of the final RM-state
- After termination of the design process the requirements in the final RM-state have been declared sufficient by the stakeholder at some point during the process

- If one of the design process objectives is that the design process should be fast and cheap, then any design object description generated during the process solely consists of standard components

In this section a number of such dynamic properties, expressed as TTL statements, are presented. These properties as listed are relevant to be considered and checked for a design reasoning trace. They need not be satisfied by all design reasoning traces; they may be used to distinguish between different types of design reasoning traces as well.

GP1 Local Requirement Satisfaction
Eventually there is a DOD that contains a satisfactory component for each local requirement that exists at that moment. Formalisation:

$\exists t\ \exists d{:}DOD\ \ state(\gamma,t,DM\text{-}state) \mid\!\!=\!\!= current\_DOD(d) \wedge$
$\quad \forall r{:}localreq\ [state(\gamma,t,RM\text{-}state) \mid\!\!=\!\!= is\_a\_current\_requirement(r) \Rightarrow$
$\quad\quad \exists\ c{:}component\ state(\gamma,t,DM\text{-}state) \mid\!\!=\!\!= part\_of\_DOD(c, d) \wedge$
$\quad\quad state(\gamma,t,DM\text{-}state) \mid\!\!=\!\!= holds\_for(r, c)]$

GP2 DM Successfulness
For each local requirement, if there is a component that satisfies it, then such a component will be added to the DOD. Formalisation:

$\forall t\ \forall r\ \forall{:}localreq\ \forall c{:}component$
$state(\gamma,t,DM\text{-}state) \mid\!\!=\!\!= is\_a\_current\_requirement(r) \wedge$
$state(\gamma,t,DM\text{-}state) \mid\!\!=\!\!= holds\_for(r, c) \Rightarrow$
$\quad \exists t' \geq t\ \exists d{:}DOD\ \exists c'{:}component\ state(\gamma,t',DM\text{-}state) \mid\!\!=\!\!= part\_of\_DOD(c', d) \wedge$
$\quad state(\gamma,t',DM\text{-}state) \mid\!\!=\!\!= holds\_for(r, c')$

GP3 RM Successfulness
At a certain point in time, all nonlocal requirements will be refined. Formalisation:

$\exists t\ \forall n{:}nonlocalreq\ \ state(\gamma,t,RM\text{-}state) \mid\!\!=\!\!= is\_a\_current\_requirement(n) \Rightarrow$
$\quad state(\gamma,t,RM\text{-}state) \mid\!\!=\!\!= requirement\_refined(n)$

The global properties presented above have been checked automatically against the simulation trace discussed in Section 6. They all turned out to hold, which confirms the fact that the simulated design processes satisfied the desired properties such as termination and successfulness.

In addition to the above, logical relationships can be and have been identified between dynamic properties at different abstraction levels. Such relationships relate the Global Properties presented in this section to some of the Local Properties presented in Section 6. They can be specified by means of logical implications or graphically by means of AND/OR trees. In these relationships, also properties at an intermediate level of aggregation (*Intermediate Properties*) occur, addressing smaller steps than Global Properties do, but bigger steps than Local Properties do. In combination with the automated checks described above, the interlevel relationships can play an important role in the analysis of design processes, because of their hierarchical structure. I.e., if a certain Global Property turns out not to hold for a given design process trace, then the table of logical relationships can be consulted in order to pinpoint which local properties are candidates for causing the failure.

8. Conclusion

In order to develop automated support for the dynamics of nontrivial design processes, the challenge of modelling and analysing such dynamics in a formal manner has to be addressed; cf. [1], [5], [6]. This paper offers an approach to do so. The complex dynamics of a design process has been analysed in such a precise way that properties of the process as a whole can be specified and, moreover, part of the analysis contains enough detail to

allow for simulation. The result of simulation has been checked against the properties of the design process as a whole.

Compared to the references mentioned above, the approach put forward is a declarative, logical approach supported by a formal language TTL for specification of dynamic properties of design processes, which has a high expressivity. Furthermore, also simulation models are specified in a declarative, logical manner, which allows using these specifications in logical analysis as well.

The paper shows the potential of this formal analysis as a technique for analysis at a high level of abstraction, and for constructing simulations at an abstract level to experiment with dynamics of a design process. The simulation actually is entailed by the analysis and requires no additional programming.

The analysis approach that is for the first time applied to design processes here, has previously been applied to complex and dynamic reasoning processes other than design, such as reasoning based on multiple representations [3]. In these cases in addition to simulated traces, also empirical (human) reasoning traces have been formally analysed. For further research it is planned to formally analyse protocols of human design processes in a similar manner.

References

[1] Baldwin and Chung (1995). A Formal Approach to Managing Design Processes. IEEE Computer, Feb. 1995, pp. 54-63.

[2] Bosse, T., Jonker, C.M., and Treur, J., (2003). Requirements Analysis in Design of Agent Systems. Vrije Universiteit Amsterdam, Department of Artificial Intelligence. Technical Report. URL: http://www.cs.vu.nl/~wai/Papers/BDBCh3.pdf

[3] Bosse, T., Jonker, C.M., and Treur, J., (2003). Simulation and analysis of controlled multi-representational reasoning processes. *Proc. of the Fifth International Conference on Cognitive Modelling, ICCM'03*. Universitats-Verlag Bamberg, 2003, pp. 27-32.

[4] Brazier F.M.T., Langen P.H.G. van, Treur J., (1996). A logical theory of design. In: J.S. Gero (ed.), *Advances in Formal Design Methods for CAD, Proc. of the Second International Workshop on Formal Methods in Design*. Chapman & Hall, New York, 1996, pp. 243-266.

[5] Brown, D. C., and Chandrasekaran, B., (1989). Design Problem Solving: Knowledge Structures and Control Strategies, Pitman, London.

[6] Corkill, D.D. (2000). When Workflow Doesn't Work: Issues in managing dynamic processes, *Proceedings of the Design Project Support using Process Models Workshop*, Sixth International Conference on Artificial Intelligence in Design, Worcester, Massachusetts, June 2000, pp. 1-13.

[7] Jonker, C.M., and Treur, J. (2002). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

[8] Jonker, C.M., Treur, J., and Wijngaards, W.C.A. (2003). A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 191-210.

PART III NEGOTIATION

CHAPTER 8

Experiments in Human Multi-Issue Negotiation: Analysis and Support

Part of this chapter will appear as Bosse, T., Jonker, C.M., Meij, L. van der, Robu, V., and Treur, J. (2005). A System for Analysis of Multi-Issue Negotiation. In: Calisti, M., Klusch, M., and Unland, R. (eds.), *Software Agent-Based Applications, Platforms and Development Kits*, Birkhaeuser Publishing Company.

Part of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2004). Experiments in Human Multi-Issue Negotiation: Analysis and Support. In: Jennings, N.R., Sierra, C., Sonenberg, L., and Tambe, M. (eds.), *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'04*, ACM Press, pp. 672-679.

Experiments in Human Multi-Issue Negotiation: Analysis and Support

Tibor Bosse[1], Catholijn M. Jonker[1], and Jan Treur[1,2]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, jonker, treur}@cs.vu.nl
http://www.cs.vu.nl/~{tbosse, jonker, treur}
[2] Universiteit Utrecht, Department of Philosophy,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Abstract. The purpose of this paper is to report on experiments in (human) multi-issue negotiation and their analysis, and to present a generic software environment supporting such an analysis. First, the paper presents a System for Analysis of Multi-Issue Negotiation (SAMIN). SAMIN is designed to analyse negotiation processes between human negotiators, between human and software agents, and between software agents. The user can enter any formal property deemed useful into the system and use the system to automatically check this property in given negotiation traces. Second, the paper presents the results of applying SAMIN in the analysis of empirical traces obtained from an experiment in multi-issue negotiation about second hand cars. In the experiment the efforts of 74 humans negotiating against each other have been analysed using SAMIN.

1. Introduction

Negotiation is a process by which a joint decision is made by two or more parties [9]. Typically each party starts a negotiation by offering the most preferred solution from the individual area of interest. If an offer is not acceptable by the other parties they make counter-offers in order to move them closer to an agreement. The field of negotiation can be split into different categories, e.g. along the following lines:

- one-to-one versus more than two parties.
- single- versus multi-issues
- closed versus open
- mediator-based versus mediator-free

The research reported in this article concerns one-to-one, multi-issue, closed, mediator-free negotiation. For more information on negotiations between more than two parties (e.g., in auctions), the reader is referred to, e.g., [12]. In single-issue negotiation, the negotiation focuses on one aspect only of the concept under negotiation. For example, a typical issue is the price of a product. Multi-issue negotiation (also called multi-attribute negotiation in the literature) is often seen a more cooperative form of negotiation, since often an outcome exists that brings joint gains for both parties, see [10].

Closed negotiation means that no information regarding preferences is exchanged between the negotiators. The only information exchanged is formed by the bids. In completely open negotiation, the negotiators honestly tell each other all their preferences, giving each other full insight in the room there is for negotiation. Raiffa [10] is stout advocate of open negotiations, arguing that the best outcome for both parties can only be

guaranteed in open negotiations. Realising that this requires a basic trust in the other parties, a compromise can be found using mediators trusted by all parties. Furthermore, much research is being done on partially open negotiations, where some information is shared during the negotiation process. More information on those negotiations can be found, e.g., in [7], [10]. However, many researchers are interested in closed negotiation, because they feel that the trust necessary for open or even partially open negotiations is often not available. The research reported in this article focusses on closed negotiation.

The use of mediators is a well-recognised tool to help the involved parties in their negotiations, see e.g., [6], [10]. The mediator tries to find a deal that is fair to all parties, and thus tries to compromise between the interest of all parties involved. The examples in politics are many-fold, and, presumably, need no further explication. On the other hand, often negotiations are performed without mediators. The reason for that can be the lack of a mediator trusted enough by the different parties, the costs of involving a mediator, or the hope of doing better than fair.

The literature on closed, multi-issue, one-to-one negotiation without mediators covers both systems to (partially) automate the negotiation process, and more analytic research focused on properties of the negotiation process and negotiation space, see Section 9. Based on that literature study and on our own analysis, a number of properties are presented here that focus largely on the dynamics of the negotiation process itself and on the results of the negotiation.

The SAMIN system presented in this paper has been developed to formally analyse such negotiation processes, i.e., multi-issue, closed, one-to-one negotiations without mediators. Basically, the system needs three different inputs:

a) a negotiation *trace* (or a set of traces)

b) a set of *dynamic properties* that are considered relevant for the negotiation process

c) the *negotiation profiles* of the participants

A *trace* is a sequence of bids by the negotiators. A *dynamic property* is an (informal, semi-formal or formal) expression that can or cannot hold for a certain trace. An example of a simple dynamic property is bid-alternation, i.e., after communicating a bid to another agent, the agent remains silent until it has received a new bid from the other agent. A *negotiation profile* is a description of the preferences of the agent within the particular negotiation domain. The profiles together define the space of possible and efficient outcomes and are, therefore, essential for the creation of a complete analysis of the performance of a negotiator. SAMIN can check automatically whether selected properties hold for the traces under analysis. Such an analysis provides a means to improve bidding strategies and bidding protocols, both for human negotiators and for software agents in automated negotiation systems.

In Section 2 formalisation of negotiation process dynamics will be discussed in terms of negotiation states, transitions, and traces. Section 3 explains the formal specification of dynamic properties. Section 4 provides example dynamic properties relevant for closed multi-issue one-to-one negotiations. SAMIN's architecture is presented in Section 5, and some details of the current prototype are presented in Section 6. The set-up of some experiments in human multi-issue negotiation is described in Section 7, and the results are provided in Section 8. Section 9 compares our work with the literature. Finally, Section 10 provides conclusions and some planned future work.

2. Formalising Negotiation Process Dynamics

Negotiation is essentially a dynamic process. To analyse those dymamics, it is, therefore, relevant to formalise and study dynamic properties of such processes. For example, how does a bid at a certain point in time compare to bids at previous time points? The formalisation introduced in this section is based on the notion of negotiation process state, negotiation transition and negotiation trace.

2.1. Formalising States of a Negotiation Process

The state of a (one-to-one) negotiation process at a certain time point can be described as a combined state consisting of two states for each of the negotiating agents: S = < S1, S2 > with:

- S1 state of agent A
- S2 state of agent B

Each of these states include the following information:

- the agent's own most recent bid
- its evaluation of its own most recent bid
- its evaluation of the other agent's most recent bid
- the history of bids from both sides and evaluations

To describe negotiation states a state ontology Ont is used. Example elements of this ontology are a sort BID for bids, and relations such as util(A, b, v) expressing that A's overall evaluation of bid b is v. Based on this ontology the set of ground atoms At(Ont) can be defined. A state is formalised as any truth assignment: At(Ont) $\rightarrow$ {t, f} to this set of ground atoms. The set of all states described by this ontology is denoted by States(Ont).

2.2. Negotiation Transitions

A particular negotiation process shows a sequence of transitions from one state S from States(Ont) to another (next) state S' from States(Ont). A transition S $\rightarrow$ S' from a state S to S' can be classified according to which agents are involved. During such a transition each of the main state components (S1, S2) of the overall state S may change. The simplest types of transition involve a single component transition. For example, when one agent generates a bid, while the other agents is just waiting: a transition of type S1 $\rightarrow$ S1 or S2 $\rightarrow$ S2. Next come transition types where both components are involved. For example, when a communication from agent A to agent B takes place, changing the state S2 of agent B: a transition of type S1 x S2 $\rightarrow$ S2. Notice that in principle, also more complex transition types are possible, involving changes of both state components at the same time, i.e., S1 x S2 $\rightarrow$ S1 x S2. In organised cooperations between multiple agents the complexity of the types of transitions is often limited by regulation of the organisation. For example, in organised negotiation processes, usually it is assumed in the protocol that after communicating a bid to the other agent, the agent remains silent until it has received a new bid from the other agent (see the dynamic property 'bid alternation' in Sections 3 and further below). Such an assumption about the protocol implies that the transitions involved in the negotiation are only of the simpler types mentioned above.

2.3. Negotiation Traces

Negotiation traces are time-indexed sequences of negotiation states, where each successive pair of states is a negotiation transition. To describe such sequences a fixed *time frame* T is assumed which is linearly ordered. A *trace* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow STATES(Ont)$, i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering.

3. Specifying Dynamic Properties of a Negotiation Process

Specification of dynamic properties of a negotiation process can be done in order to *analyse* its dynamics, for example to find out how certain properties of a negotiation process as a whole relate to properties of a certain subprocess, or to verify or evaluate a negotiation model. To formally specify dynamic properties that express characteristics of dynamic processes (such as negotiation) from a temporal perspective an expressive language is needed. To this end the *Temporal Trace Language* TTL is used as a tool; cf. [5], which is briefly defined as follows.

The Language TTL for Dynamic Properties

The set of dynamic properties DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, a certain state of the agent A during a negotiation process at time point t is indicated by

state(γ, t, A).

In the third argument, instead of A also specific parts of A can be used, such as input(A), or output(A). These state indicators can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus:

state(γ, t, A) |= p

denotes that state property p holds in trace γ at time t in the state of agent A. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts T for time points, Traces for traces and F for state formulae, using quantifiers and the usual first-order logical connectives such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$.

As an example, consider the dynamic property bid alternation, which states that for all two different moments in time t1, t3, that A generates a bid, there is a moment in time t2, with t1 < t2 < t3, such that A received a bid generated by B. In formal TTL-format, this property is expressed as:

bid_alternation(γ:TRACE) $\equiv$
$\forall$ A, B: AGENT, $\forall$ b1, b3: BID, $\forall$ t1, t3:
t1 < t3 &
state(γ, t1, output(A)) |== to_be_communicated_to_by(b1, B, A) &
state(γ, t3, output(A)) |== to_be_communicated_to_by(b3, B, A)
$\Rightarrow$ $\exists$b2, $\exists$t2: t1 < t2 < t3 &
state(γ, t2, input(A)) |== communicated_to_by(b2, A, B)

Usually for reasons of presentation dynamic properties are expressed in informal or semi-formal forms. In the next section, an overview of different types of dynamic properties relevant for negotiation is given.

4. Dynamic Properties of Closed Multi-Issue One-to-One Negotiation Processes

The properties relevant for analysing the dynamics of closed multi-issue one-to-one negotiation, can be divided into the following types:

- **Bid properties** give some information about a specific bid. They are usually defined in terms of the negotiation space and the profiles of the negotiators. Bid properties concern, for example, the Pareto efficiency of a bid.
- **Result properties** are a subset of the set of bid properties, concerning only the last bid of a negotiation process (i.e., the final agreement).
- **Bid comparison properties** compare two arbitrary bids with each other. An example is "Domination": a bid b1 dominates a bid b2 with respect to agents A and B iff both agents prefer bid b1 over bid b2; see below for a formalisation. Another example is "Better Social Welfare": the social welfare of bid b1 is better than that of bid b2 with respect to agents A and B iff the sum of the utility values of bid b1 is bigger than the sum of the utility values of bid b2 [6], [10].
- **Step properties** are a subset of the set of bid comparison properties, concerning only the transitions between successive bids. Hence, they are restricted to the combinations of bids of one party that directly follow each other. A range of step properties can be defined to express that an agent varies its proposals to a certain extent, over a number of different issues, in a certain time interval, with respect to utility, and so on.
- **Limited interval properties** concern parts of traces. Basically, they state that each step in a certain interval satisfies a certain step property. For instance: a negotiation process is Pareto-monotonous for the interval [t1, t2] iff for all successive bids b1, b2 in the interval b2 dominates b1 (see below).
- **Trace properties** are a subset of the set of limited interval properties, concerning whole traces.
- **Multi-trace properties** compare the dynamics observed in more than one trace. An example is "Better Negotiator": agent A is a better negotiator than agent B iff in more than 60% of the negotiations between A and B, the deal reached is more to the advantage of agent A than of agent B. Another example is "Improved Negotiation Skill": Agent A improves its negotiation skill iff after having negotiated a number of times in comparable settings against comparable opponents, the deals made by A are more to A's advantage than in the first negotiations.
- **Protocol properties** are discussed quite extensively in literature, and basically fall into two categories: *Bid validity* and *Timeouts*. Bid validity entails that the bid is offered at an appropriate moment, and that it satisfies constraints on their value [8]. A specific instance is: "over time the bids of negotiators A and B alternate". Timeouts determine the closing of the negotiation [8]. An instance of such a property is the following: "for both negotiators A and B, unless the stop criterion

holds, a new proposal is generated by A upon receival of a proposal by B." The stop criterion holds for agent A at time t, if at time t agent A receives a bid by negotiation partner B that is at least as good as the last bid made by A.

Note that the first two types are basically *static properties*, whereas the other types are *dynamic properties*: they specify behaviour over time. In Appendix A for each of these types a number of properties are described in detail, both in informal and in formal notation. In this section, only a small selection of relevant properties is presented.

configuration_differs(b1:BID, b2:BID) ≡
∃a: ISSUE, ∃v1, v2: VALUE:
value_of(b1, a, v1) & value_of(b2, a, v2) & v1 ≠ v2

This bid comparison property states that two bids b1 and b2 differ in configuration iff there is an issue that has a different value in both bids. Similar properties can be defined stating that two bids differ in configuration in at least x issues. This property can also be used as a building block to specify step properties (see the property agent_views_agent_makes_config_variation below), limited interval properties, and trace properties.

strictly_dominates(b1:BID, b2:BID, A:AGENT, B:AGENT) ≡
∀vA1, vA2, vB1, vB2 : real :
util(A, b1, vA1) & util(A, b2, vA2) & util(B, b1, vB1) & util(B, b2, vB2) ⇒ vA1 > vA2 & vB1 > vB2

This bid comparison property states that a bid b1 dominates a bid b2 with respect to agents A and B iff both agents prefer bid b1 over bid b2. This notion is related to Pareto Efficiency, see e.g., [10]. The property could also be changed to weakly_dominates by changing the > sign into the ≥ sign. Moreover, it can be used as a building block to specify step properties, limited interval properties (see strict_pareto_monotony below), and trace properties.

agent_consecutively_bids_to(γ:trace, A:AGENT, t1:time, b1:BID, t2:time, b2:BID, B:AGENT) ≡
state(γ, t1, output(A)) |== to_be_communicated_to_by(b1, A, B) &
state(γ, t2, output(A)) |== to_be_communicated_to_by(b2, A, B) &
t1 < t2 &
[∀t3, ∀b3: BID:
t1 < t3 < t2 ⇒ state(γ, t3, output(A)) |=/= to_be_communicated_to_by(b3, A, B)]

This step property states that in a negotiation process γ agent A consecutively bids b1 at time t1 and then b2 at time t2 to agent B. Together with bid comparison properties like configuration_differs and strictly_dominates, this property can be used as a building block to specify other step properties (see agent_views_agent_makes_config_variation below), limited interval properties, and trace properties.

is_followed_by(γ:trace, A:AGENT, t1:time, b1:BID, B:AGENT, t2:time, b2:BID) ≡
state(γ, t1, output(A)) |== to_be_communicated_to_by(b1, A, B) &
state(γ, t2, output(B)) |== to_be_communicated_to_by(b2, B, A) &
t1 < t2 &
[∀t3, ∀C, D: AGENT, ∀b3: BID:
t1 < t3 < t2 ⇒ state(γ, t3, output(C)) |=/= to_be_communicated_to_by(b3, C, D)]

This step property states that in a negotiation process γ a bid b1 at time t1 is followed by a bid b2 at time t2 iff bids b1 and b2 are subsequent bids in γ. The difference with the above

step property is that here it is demanded that the consecutive bids are made by different agents instead of the same agent. Together with bid comparison properties like configuration_differs and strictly_dominates, this property can be used as a building block to specify other step properties, limited interval properties (see strict_pareto_monotony below), and trace properties.

agent_views_agent_makes_config_variation(γ:trace, A:AGENT, B:AGENT, t1:time, b1:BID, t2:time, b2:BID) ≡
agent_consecutively_bids_to(γ, A, t1, b1, t2, b2, B) &
configuration_differs(b1, b2) &
∀vA1, vA2 : real :
util(A, b1, vA1) & util(A, b2, vA2) ⇒
vA1 = vA2

This step property makes use of the previous properties configuration_differs and agent_consecutively_bids_to. It states that, for a pair of consecutive bids in trace γ, in the view of agent A, agent B varies the configuration, but not the utility. Note that one agent can both be agent A and B, or A and B can refer to different agents. Property like this could be useful to find out what kind of opponent the negotiator is dealing with.

strict_pareto_monotony(γ:trace, tb:time, te:time) ≡
∀t1, t2, ∀A, B: AGENT, ∀b1, b2: BID :
[tb ≤ t1 < t2 ≤ te & is_followed_by(γ, A, t1, b1, B, t2, b2)]
⇒ strictly_dominates(b2, b1, A, B)

This limited interval property makes use of the previous properties strictly_dominates and is_followed_by. It states that a negotiation process γ is strictly Pareto-monotonous for the interval [t1, t2] iff for all successive bids b1, b2 in the interval b2 dominates b1. By choosing tb and te in an appropriate way it can be transformed into a trace property. Generally, traces that satisfy this property are not abundant in (human) real world multi-issue negotiations, since if the profiles of the two parties are strongly opposed (with emphasis on the same issues), even in multi-issue situations a gain for the one often implies a loss for the other. If, however, the profiles are less opposed, pareto-monotony may occur.

pareto_inefficiency(b:BID, A:AGENT, B:AGENT, ε:real) ≡
∀vA, vB : real :
util(A, b, vA) & util(B, b, vB) ⇒ pareto_distance(vA, vB) = ε

This bid property informally states that with respect to agents A and B, the Pareto inefficiency of a bid b is the number ε that indicates the distance to the Pareto Efficient Frontier according to some distance measure d in utilities. Here, d(b1, b2) is the distance between the bids b1 and b2 when viewed as points in the plane of utilities. The function to measure the distance in the plane can still be filled in, e.g., the sum of absolute differences of coordinates, or the square root of the sum of squares of the differences, or the maximum of the differences of the coordinates. The Pareto Efficient Frontier is the set of all bids b for which there is no other bid b' that dominates b. Hence, in case the Pareto Inefficiency of a bid is 0, there is no other bid that dominates it. By filling in the resulting agreement of a negotiation for bid b, the property is transformed into a result property. In general, determining the number ε for which this property holds is a good measure for checking the success of the negotiation process. In a similar way, the property nash_inefficiency can be formulated, which calculates the distance from a certain bid to the Nash Point. This is the point (on the Pareto Efficient Frontier) for which the product of both utilities is maximal, see e.g., [10].

5. Design of the SAMIN architecture

SAMIN is a software environment that has been designed at the Vrije Universiteit for the analysis of multi-issue negotiation processes. This section describes the role SAMIN can take in an analysis setting of negotiation processes, and presents the global design of the architecture chosen for SAMIN. In Section 6 the parts of this design that have been implemented in the current SAMIN prototype are described in more detail.

The SAMIN system has been designed to work together in interaction with a human analyst and either human or software agent negotiators. As depicted in Figure 1, the analyst determines the properties that SAMIN is to use in the analysis of negotiation processes. He or she can select (and if necessary adapt) properties from SAMIN's library, or can construct new properties with the help of SAMIN's special dynamic property editor. SAMIN can only analyse a negotiation process if it has access to the profiles used by the different parties, and the bids exchanged between the parties. SAMIN does not influence the negotiation while it is being carried out, it only observes either during the negotiation, or afterwards.

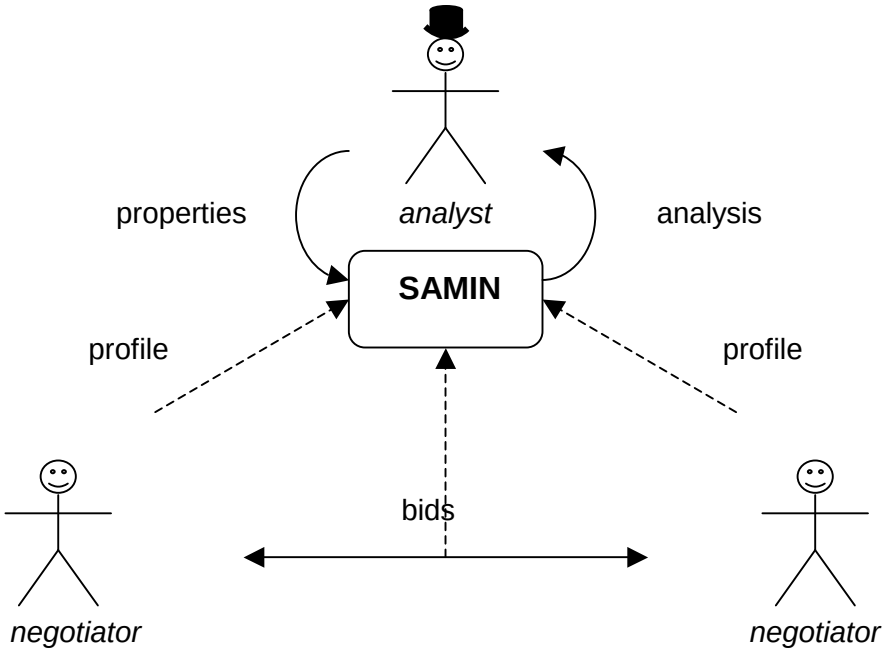


Figure 1. SAMIN in its environment

The analysis result of one or more negotiations is presented to the human analyst. The analyst can use that for cognitive scientific purposes, to train human negotiators, or to improve the strategies of software agents. Interesting for the future might be to present the results directly after the conclusion of the negotiation to a software agent negotiator that is capable of learning so that the agent can use the result to improve its negotiation skill by itself. A negotiation process can be monitored directly by SAMIN (if the agents allow interfacing), or the negotiation trace can be written to a file and be analysed in hindsight by SAMIN. The current version of SAMIN is developed especially for closed multi-issue one-to-one negotiations, entailing that the only information exchanged between the negotiators are the bids.

The input required by SAMIN, see Figure 1, consists of properties, profiles, and traces of bids. Its output consists of an analysis. As mentioned before, SAMIN offers the user both a library of properties to choose from and a dynamic property editor to create new

properties. Profiles can be obtained in two ways. Either the negotiator presents a pre-specified profile to SAMIN or the negotiator can use SAMIN's interactive profile editor to create it in SAMIN. Pre-specified profiles have to be in a format recognised by SAMIN (see also Section 6.1). The trace of bids required by SAMIN can be obtained by SAMIN monitoring the bids exchanged between the negotiators during the negotiation process. This only requires the bids to be in a format recognised by SAMIN and the possibility to "overhear" the communication between the negotiators. Another possibility is that the bids exchanged during a negotiation process are stored in a special file. If the bid-traces are in the right format, SAMIN can perform analysis on one or on a combination of such traces after the negotiation has been completed. If the negotiators wish to do so, they can use SAMIN's bid ontology editor to define what a bid should look like, before entering the negotiation phase. Construction of a bid ontology and the profiles is part of the pre-negotiation phase [10].

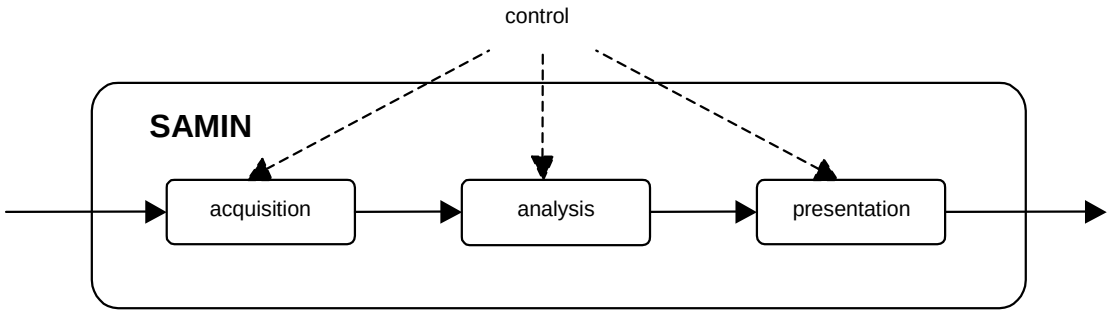


Figure 2. Global Design of the SAMIN architecture

For a global overview of the design of the SAMIN architecture, see Figure 2. It consists of components to acquire the input necessary for analysis, to perform the analysis, and to present the results of the analysis. Furthermore, SAMIN maintains a library of properties, templates of properties, bid ontologies, and profile ontologies.

6. The SAMIN prototype

Within the current SAMIN prototype a number of the components of the SAMIN architecture have been implemented. These implemented components will be briefly described below.

6.1. The acquisition component

The acquisition component is used to obtain the required input to perform analysis. It consists of an *ontology editor*, a *dynamic property editor* and a *trace determinator*.

The ontology editor is used for the construction of bid ontologies and profile ontologies necessary to automatically interpret the bids exchanged by the negotiators, and to automatically interpret the profiles of the negotiators. The ontology editor is typically used to construct a bid ontology and a profile ontology, thus allowing the user to identify the issues to be negotiated, the values that each of these issues can take, and the structure of bids, in the bid ontology. Furthermore, in specifying the profile ontology (that makes use of the bid ontology) the user identifies the possible evaluations that can be given to issue-value combinations, the possible interdependencies between issues, and the utility functions of bids.

The dynamic property editor based on TTL supports the gradual formalisation of dynamic properties that are initially entered in natural language (informal). It is also possible to directly enter formal properties, see Figure 3.

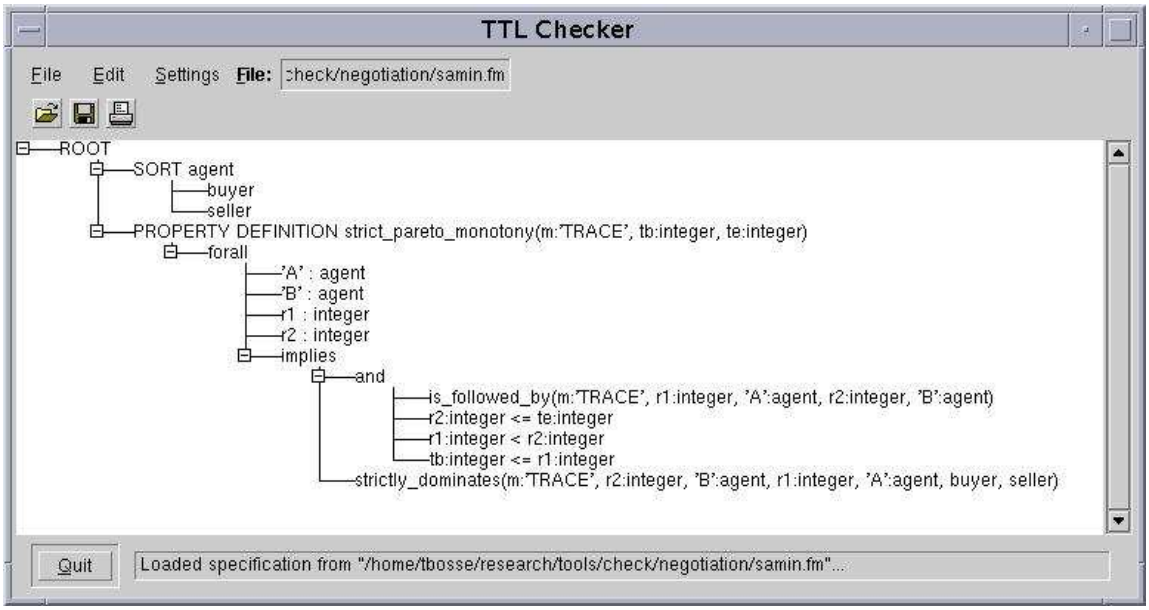


Figure 3. Dynamic Property Editor

The trace determinator can be used interactively with the analyst to determine what traces to use in the analysis. The user can interactively locate the files containing the traces to be checked. The traces themselves can be of three categories: (human) empirical traces, simulated traces, and mixed traces. An empirical trace is the result of an existing human negotiation process. A simulated trace is the result of an automated negotiation system. A mixed trace is the result of a human negotiating with a software agent. To support the acquisition of traces of all three types, a dedicated interface has been created for SAMIN.

6.2. The analysis component

The analysis component currently consists of a *logical analyser* that is capable of checking whether a dynamic property holds for a trace, or for a number of traces. If a dynamic property does not hold in a trace, then the software reports the places in the trace where the property failed.

6.3. The presentation component

The presentation component currently includes a tool that visualises the negotiation space in terms of the utilities of both negotiators. This *visualisation tool* plots the bid trajectory in a 2-dimensional plane, see Figure 4.

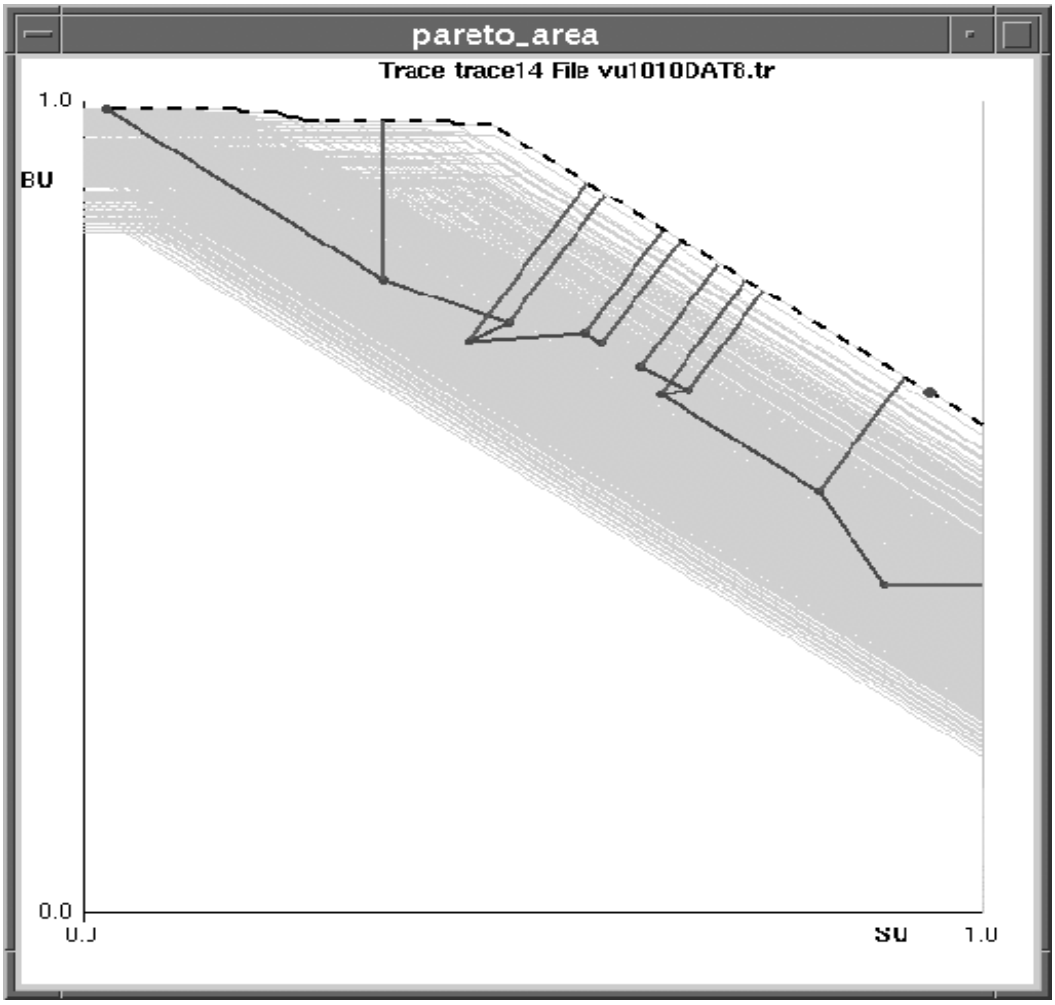


Figure 4. Visualisation Tool

In this Figure, the seller's utility of a bid is on the horizontal axis, and the buyer's utility is on the vertical axis. The light area corresponds to the space of possible bids. In this area, each curve is a continuous line, corresponding to a different combination of discrete issues. The specific position on the line is determined by the continuous issue 'price'. Since in this particular domain 4 discrete issues with 5 possible values occur (see next Section), there are already 625 (= 5^4) different curves. In Figure 4, the sequences of actual bids made by both buyer (left) and seller (right) are indicated by the two dark angular lines. The dotted line indicates the Pareto Efficient Frontier according to the profiles of the negotiating agents, and the short dark lines show the distance from each bid to this frontier. The small dot that is plotted on the Pareto Efficient Frontier (on the right) corresponds to the Nash Point. From this picture, it is clear that both negotiators make more and more concessions over time. Eventually, they reach a point that does not lie on the Pareto Efficient Frontier, but is rather close to it anyhow.

7. Design of the Human Multi-Issue Negotiation Experiments

To illustrate the use of analysing multi-issue negotiation processes, SAMIN has been applied in a case study. As mentioned in Section 6.2, the analysis component of SAMIN takes traces and formally specified dynamic properties as input and checks whether a property holds for a trace. Using automatic checks of this kind, some of the properties

provided in Section 4 have been checked against empirical traces generated by students during Practical Sessions in Multi-Issue Negotiation. The domain of the case study, a negotiation about second hand cars, will be presented in detail in Section 7.1. Section 7.2 describes the setup of the experiments performed in the case study. The results of the analysis of the acquired traces will be shown in Section 8.

7.1. Domain: second hand cars

The object of negotiation is a particular second hand car. Notice that here the object of the negotiation is already fixed, but that this is not necessarily the case. As an alternative, the specific car to be sold could be negotiation upon as well. In multi-issue negotiation, a bid has the form of values assigned to a number of issues of the object under negotiation. Within this domain, the relevant issues are cd_player, extra_speakers, airco, drawing_hook and price. Consequently, a bid consists of an indication of which CD player is meant, which extra speakers, airco and drawing hook, and what the price of the bid is. The goal of the negotiators is to find agreement upon the values of the four accessories and the price. Here, the price issue has a continuous value, whilst the other four issues have a discrete value from the set {good, fairly_good, standard, meager, none}. These values are assumed to be objective indicators from a consumer organisation, so there can be no discussion about whether a certain CD player is good or fairly good.

Before the negotiation starts, both parties specify their *negotiation profile*: for all issues with discrete values they have to assign a number to each value, indicating how satisfied they would be with that particular value for the issue (e.g. "I would be very happy to buy/sell a good CD player, a bit less happy with a fairly good CD player, …" and so on). The buyer also has to indicate what is the maximum amount of money (s)he would be willing to spend. Moreover, both parties have to assign a number to each of the issues, indicating how important they judge that issue (e.g. "I don't care that much which CD player I will buy/sell"). Notice that this does not conflict with the above statements. An example negotiation profile for a buyer is shown in Figure 5. In addition to this negotiation profile, the seller is also provided with a *financial profile*. This is a list of all issues, in which for each issue it is indicated how much it costs, both to buy it and to build it into the car. Since we focus on closed negotiation, none of the profiles will be available for the other negotiator. However, SAMIN has access to both profiles.

When both parties have completed their profiles, the negotiation starts. To help human negotiators generating their bids, the system offers a special tool that calculates the utility of a bid before it is passed to the opponent. The utility U_B of a bid B is defined by the weighted sum over the issue evaluation values $E_{B,j}$ for the different issues denoted by:

$$U_B = \Sigma_j w_j \; E_{B,j}$$

The weight factors w_j are based on the attribute importance factors. Here scaling takes place (the sum of weight factors is made 1, and the evaluation values $E_{B,j}$ are between 0 and 1) so that the utility is indeed is between 0 and 1; for more details, see [4]. Since the negotiators have individual negotiation profiles, for each bid the seller's utility of the bid is different from the buyer's utility of the bid.

Besides for facilitating the bidding process, the profiles are used by SAMIN to analyse the resulting traces (see Section 8). For example, to check whether the property Pareto-Monotony holds (i.e., "For each combination of successive bids b1, b2 in the trace, both agents prefer bid b2 over bid b1"), the software must have a means to determine when an agent "prefers" one bid over another.

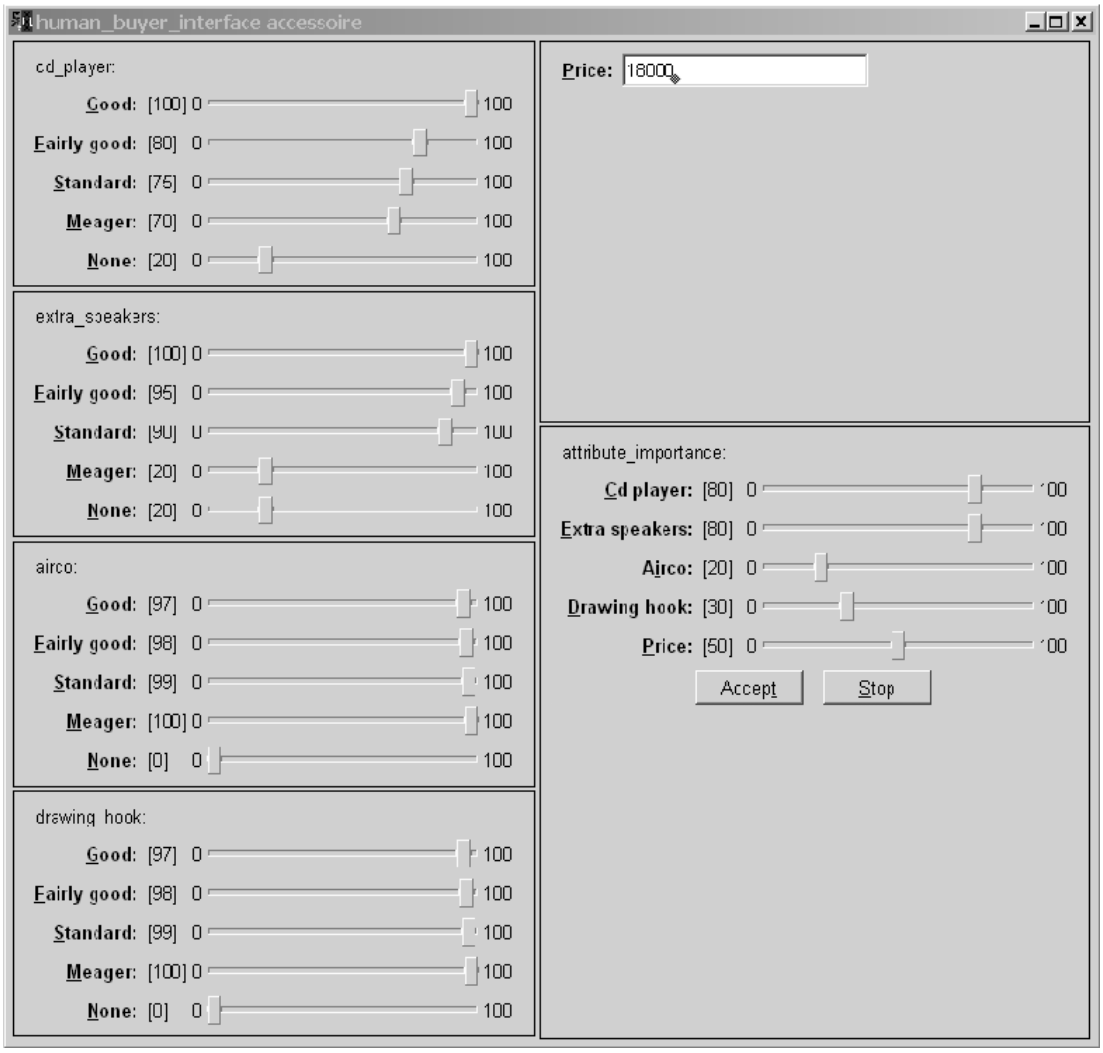


Figure 5. Example Buyer's Negotiation Profile

7.2. Experimental Setup

Participants. Seventy-four subjects participated in the experiment, in three different sessions. All sessions took place during a master class for students of the final classes of the VWO (a particular type of Dutch High School). The age of the students mostly was about 17 years, but varied between 14 and 18 years. Most of them were males. In the first session, in March 2002, 30 students participated. In the second session, in March 2003, 28 students participated. In the third session, in November 2003, 16 students participated.

Method. Before starting the experiment, the participants were provided some background information on negotiation, and in particular about multi-issue negotiation. Some basic negotiation strategies were discussed. In addition, the second hand car example was explained. Then they were asked to start negotiating, thereby taking a profile in mind (that had to be specified first) aiming at obtaining the best possible deal, without showing their own profile to the opponent. The negotiation process was performed using different terminals over a network, which allowed each participant to negotiate with another anonymous participant. All negotiators could input their bids within a special interface. The resulting negotiation traces were logged by the system, so that they could be re-used for the purpose of analysis. A screenshot of an example negotiation trace is depicted in

Figure 6. This trace is shown from the perspective of the buyer. In the upper part of the window, the buyer's own bids are displayed, including the buyer's utility for each bid. In the middle part, the bids of the seller are displayed, including the buyer's utility for each bid. The lower part consists of the bidding interface, which allows the buyer to input his bid and pass it to the seller.

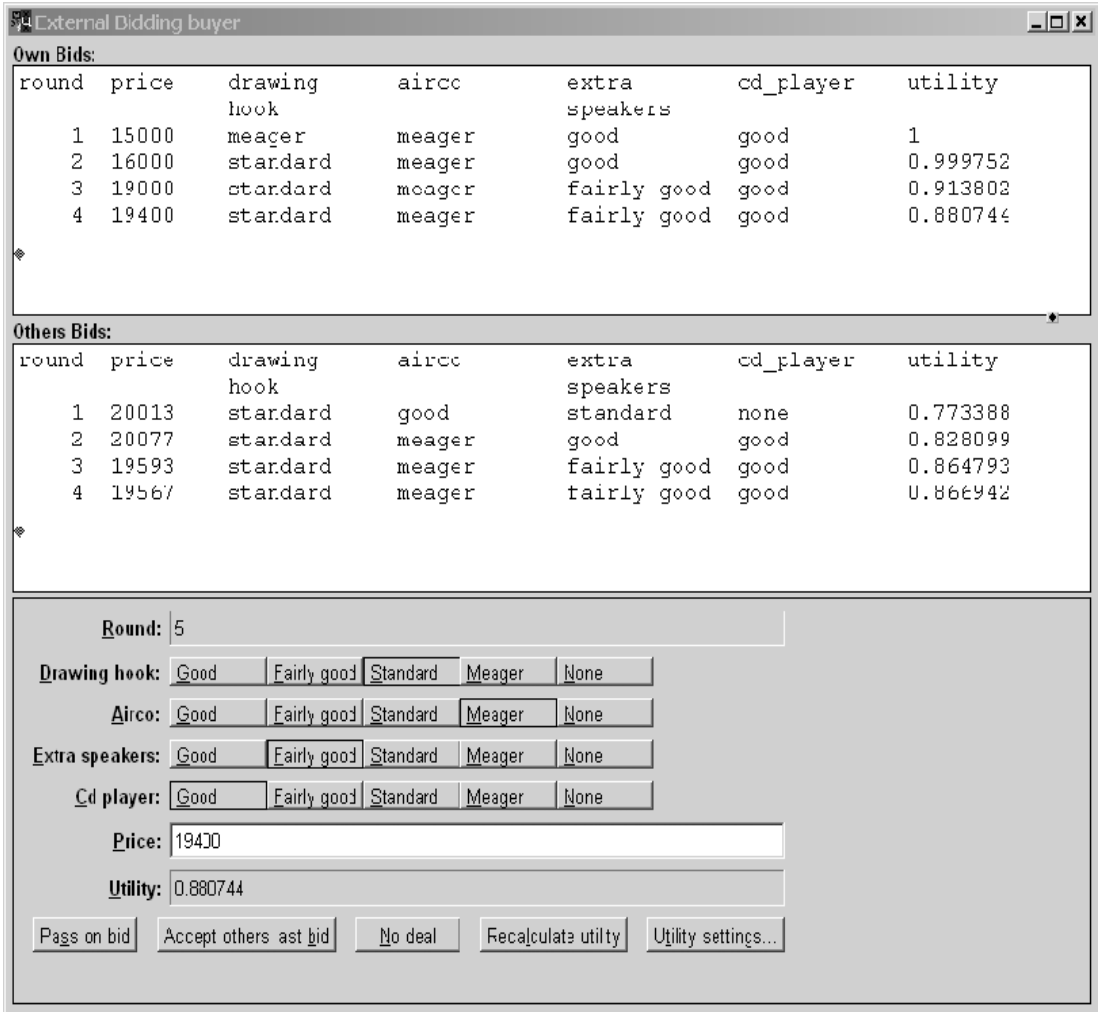


Figure 6. Example Negotiation Trace

8. Results of the Human Experiments

Using the SAMIN prototype, a number of relevant dynamic properties for multi-issue negotiation (also see Section 4) have been checked against the traces that resulted from the experiments. In this Section, a selection of interesting results is reported:

Obviously, the property bid_alternation (Section 3) holds for all traces. This means that all participants have committed to the protocol, which prescribes that as long as the negotiation lasts, a bid from A to B should be followed by a bid from B to A.

In none of the traces, the Pareto_inefficiency (Section 4) of the resulting deal was equal to 0. In several cases, during the negotiation some bids made by one of both parties temporarily lay on the Pareto Efficient Frontier, but the resulting bids never did. On average, the negotiating agents performed only slightly above halfway, i.e., the resulting bids lay somewhat above the middle of the space of possible bids (the light area of Figure

4). Apparently, it is difficult for human negotiators to guess the Pareto Inefficiency. As a result, they find it hard to decide what is the right moment to accept a proposal.

As can be derived from the previous conclusion, also the Nash Point was never reached in any final agreement, nor was it reached during any of the negotiations.

When used as a trace property, the property strict_Pareto_monotony did not hold in any of the traces. When used as a limited interval property, it sometimes held during a very short interval, but hardly ever during more than three steps. Apparently, the profiles of the negotiating parties were often strongly opposed, meaning that a gain for one party implies a loss for the other. However, when changing the criterion of strict domination into weak domination, the property often held for larger intervals. Most of the time, these intervals corresponded to the "end phase" of the negotiation: the phase in which the only issue on which no agreement has yet been reached, is the price.

9. Discussion

In the literature on negotiation a number of systems are described. Sometimes it is stated what properties these systems have, sometimes not. If properties are mentioned they can be of different types, and also the justifications of them can be of different degree or type. Examples of such literature are the following.

Faratin, Sierra, and Jennings [2] concentrate on many parties, many-issues, single-encounter closed negotiations with an environment of limited resources (time among them). Agents negotiating using the model are guaranteed to converge on a solution in a number of situations. The authors do not compare the solutions found to fair solutions (Nash Equilibrium, Maximal Social Welfare, Maximal Equitability), nor whether the solutions are Pareto Efficient.

Klein, Faratin, Sayama, and Bar-Yam [6] developed a mediator-based negotiation system to show that conceding early (by both parties) often is the key to achieving good solutions. Hyder, Prietula, and Weingart [3] showed that substantiation (providing rationale for your position to persuade the other person to change their mind) interferes with the discovery of optimal agreements.

Weingart et al. [13] found that the Pareto efficiency of agreements between naïve negotiators could be significantly improved by simply providing negotiators with descriptions of both integrative and distributive tactics. Although Pareto *efficiency* was positively influenced by the tactics, Pareto *optimality* was only minimally affected.

Compared to [8], [11], [12], the properties identified in this paper are geared towards the analysis of the dynamics of the negotiation process, whereas theirs are more oriented towards the negotiation outcome, rationality and use of resources.

A previous version of SAMIN was first developed as an analysis environment for the multi-issue negotiation system ABMP, see [4]. However, the scope of SAMIN as it has been set up as a generic environment is much broader. SAMIN can be seen as a logical next step, given the existing negotiation-related systems and the existing literature on the formalisation and analysis of negotiation, to provide a bridge between such negotiation systems and the analysis of their properties.

10. Conclusions and Future Work

SAMIN, the system for analysis of multi-issue negotiation introduced here, has proved to be a valuable tool to analyse the dynamics of human-human closed negotiation against a number of dynamic properties. Our analysis shows that humans find it difficult to guess

where the Pareto Efficient Frontier is located, making it difficult for them to accept a proposal. Although humans apparently do not negotiate in a strictly Pareto-monotonous way, when considering larger intervals, a weak monotony can be discovered. Such analysis results can be useful in two different ways: to train human negotiators, or to improve the strategies of software agents.

Currently, SAMIN is being used to analyse the dynamics of humans negotiating against software agents of the ABMP system. Future research is to analyse the dynamics of other types of (e.g., more experienced) human negotiators and of automated negotiation systems and to test the effectiveness of training methods for negotiation. As a simple extension, for example, if a dynamic property checked in a trace turns out to fail (see Section 6.2 above), a more detailed analysis can be given of the part(s) of the formula that cause(s) the failure.

Finally, we plan to extend SAMIN to provide feedback to a negotiator who is in the middle of a negotiation process, where SAMIN only has access to the same information as the negotiator.

References

[1] Bosse, T., Jonker, C. and Treur, J., Formalisation of Dynamic Properties of Multi-Issue Negotiations. Vrije Universiteit Amsterdam, Department of Artificial Intelligence. Technical Report, 2004.

[2] Faratin, P., Sierra, C., and Jennings, N.R., Negotiation decision functions for autonomous agents. In: International Journal of Robotics and Autonomous Systems, vol. 24(3-4), 1998, pp. 159 – 182.

[3] Hyder, E.B., Prietula, M.J., and Weingart, L.R., Getting to Best: Efficiency versus Optimality in Negotiation, In: Cognitive Science, vol. 24 (2), 2000, pp. 169 – 204.

[4] Jonker, C.M., Treur, J., An Agent Architecture for Multi-Attribute Negotiation. In: B. Nebel (ed.), *Proceedings of the 17th International Joint Conference on AI, IJCAI'01*, 2001, pp. 1195 - 1201.

[5] Jonker, C.M., and Treur, J., Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

[6] Klein, M., Faratin, P., Sayama, H., and Bar-Yam, Y., (2001), Negotiating Complex Contracts. Paper 125 of the Center for eBusines@MIT. http://ebusiness.mit.edu.

[7] Kowalczyk, R, Bui, V., On Constraint-Based Reasoning in e-Negotiation Agents. In: Dignum, F, and Cortés, U., (eds.), *Agent-Mediated Electronic Commerce III, Current Issues in Agent-Based Electronic Commerce Systems*, Lecture Notes in Computer Science, vol. 2003, Springer – Verlag, pp. 31-46.

[8] Lomuscio, A.R., Wooldridge, M., and Jennings. N.R., (2000), A classification scheme for negotiation in electronic commerce, In: *International Journal of Group Decision and Negotiation*, vol. 12(1), January 2003.

[9] Pruitt, D.G., *Negotiation Behavior*, Academic Press, 1981.

[10] Raiffa, H., *Lectures on Negotiation Analysis*, PON Books, Program on Negotiation at Harvard Law School, 513 Pound Hall, Harvard Law School, Cambridge, Mass. 02138, 1996.

[11] Rosenschein, J.S., and Zlotkin, G., Rules of Encounter: Designing Conventions for Automated Negotiation among Computers. The MIT Press, Cambridge, MA, 1994.

[12] Sandholm, T., Distributed rational decision making, In: Weiss, G., (ed.), *Multi-agent Systems: A Modern Introduction to Distributed Artificial Intelligence*, MIT Press, 1999, pp. 201 – 258

[13] Weingart, L.R., Hyder, E.H., and Prietula, M.J., Knowledge matters: The effect of tactical descriptions on negotiation behavior and outcome. In: *Journal of Applied Psychology*, vol. 78, 1996, pp. 504-517.

Appendix A. Dynamic Properties of Negotiation Processes

bid_alternation(γ:trace)
Over time the bids of A and B alternate: thus for all two different moments in time t1, t3, that A generated a bid, there is a moment in time t2, with t1 < t2 < t3, such that A received a bid generated by B.
∀ A, B: AGENT, ∀ b1, b3: BID, ∀ t1, t3:
t1 < t3 &
state(γ, t1, output(A)) |== to_be_communicated_to_by(b1, B, A) &
state(γ, t3, output(A)) |== to_be_communicated_to_by(b3, B, A) ⇒
∃b2, ∃t2: t1 < t2 < t3 &
state(γ, t2, input(A)) |== communicated_to_by(b2, A, B)

is_followed_by(γ:trace, A:AGENT, t1:time, b1:BID, B:AGENT, t2:time, b2:BID)
In a negotiation process γ bid b1 at time t1 is followed by a bid b2 at time t2 iff bids b1 and b2 are subsequent bids in γ.
state(γ, t1, output(A)) |== to_be_communicated_to_by(b1, A, B) &
state(γ, t2, output(B)) |== to_be_communicated_to_by(b2, B, A) &
t1 < t2 &
[∀t3, ∀C, D: AGENT, ∀b3: BID:
t1 < t3 < t2 ⇒ state(γ, t3, output(C)) |=/= to_be_communicated_to_by(b3, C, D)]

agent_consecutively_bids_to(γ:trace, A:AGENT, t1:time, b1:BID, t2:time, b2:BID, B:AGENT)
In a negotiation process γ agent A consecutively bids b1 at time t1 and then b2 at time t2 to agent B.
state(γ, t1, output(A)) |== to_be_communicated_to_by(b1, A, B) &
state(γ, t2, output(A)) |== to_be_communicated_to_by(b2, A, B) &
t1 < t2 &
[∀t3, ∀b3: BID:
t1 < t3 < t2 ⇒ state(γ, t3, output(A)) |=/= to_be_communicated_to_by(b3, A, B)]

stop_criterion(γ:trace, A:AGENT, t2:time)
The stop criterion holds for agent A at time t, if at time t agent A receives a bid by negotiation partner B that is at least as good as the last bid made by A.
∃t1, ∃B: AGENT, ∃b1, b2: BID:
state(γ, t2, input(A)) |== communicated_to_by(b2, A, B) &
state(γ, t1, output(A)) |== to_be_communicated_to_by(b1, B, A) &
is_followed_by(γ, t1, b1, t2, b2) &
util(γ, A, b1) ≤ util(γ, A, b2)

negotiation_continuation(γ:trace)
For both A and B, unless the stop criterion holds, a new proposal is generated by A upon receival of a proposal by B.
∀t, ∀A, B: AGENT, ∀b1: BID:
¬stop_criterion(γ, A, t) &
state(γ, t, input(A)) |== communicated_to_by(b1, A, B) ⇒
[∃b2: BID ∃t2: t2 > t & state(γ, t2, output(A)) |== to_be_communicated_to_by(b2, B, A)]

strictly_dominates(b1:BID, b2:BID, A:AGENT, B:AGENT)
A bid b1 dominates a bid b2 with respect to agents A and B iff both agents prefer bid b1 over bid b2.
∀vA1, vA2, vB1, vB2 : real :
util(A, b1, vA1) & util(A, b2, vA2) & util(B, b1, vB1) & util(B, b2, vB2) ⇒
vA1 > vA2 & vB1 > vB2

weakly_dominates(b1:BID, b2:BID, A:AGENT, B:AGENT)
A bid b1 dominates a bid b2 with respect to agents A and B iff both agents prefer bid b1 over bid b2.
∀vA1, vA2, vB1, vB2 : real :
util(A, b1, vA1) & util(A, b2, vA2) & util(B, b1, vB1) & util(B, b2, vB2) ⇒
vA1 ≥ vA2 & vB1 ≥ vB2

strictly_better_social_welfare(b1:BID, b2:BID, A:AGENT, B:AGENT)
The social welfare of bid b1 is better than that of bid b2 with respect to agents A and B iff the sum of the utility values of bid b1 is bigger than the sum of the utility values of bid b2. See also [6,10].
∀vA1, vA2, vB1, vB2 : real :
util(A, b1, vA1) & util(A, b2, vA2) & util(B, b1, vB1) & util(B, b2, vB2) ⇒
vA1 + vB1 > vA2 + vB2

strictly_better_equitability(b1:BID, b2:BID, A:AGENT, B:AGENT)
A bid b1 has a better equitability than bid b2 with respect to agents A and B iff the difference in the utility values of bid b1 is less than the difference in utility values of bid b2.
∀vA1, vA2, vB1, vB2 : real :
util(A, b1, vA1) & util(A, b2, vA2) & util(B, b1, vB1) & util(B, b2, vB2) ⇒
| vA1 - vB1 | < | vA2 - vB2 |

ε-equitability(b:BID, A:AGENT, B:AGENT, ε:real)
A bid b has ε-equitability with respect to agents A and B iff the difference in the utility values of bid b is less than ε. Thus, a bid that has an equitability of 0 has a maximum equitability. This definition corresponds to the idea of Raiffa to maximize the minimum utility [10].
∀vA, vB : real :
util(A, b, vA) & util(B, b, vB) ⇒
| vA - vB | ≤ ε

pareto_inefficiency(b:BID, A:AGENT, B:AGENT, ε:real)
With respect to agents A and B, the Pareto inefficiency of a bid b is the number ε that indicates the distance to the Pareto Efficient Frontier according to some distance measure d in utilities. Here d(b1, b2) is the distance between the bids b1 and b2 when viewed as points in the plane of utilities.
∀vA, vB : real :
util(A, b, vA) & util(B, b, vB) ⇒
pareto_distance(vA, vB) = ε

making_global_concession(γ:trace, A:AGENT, t1:time, b1:BID, t2:time, b2:BID, B:AGENT)
In a negotiation process γ agent B makes a global concession to agent B with respect to bid b1 at time t1 and bid b2 at time t2 iff both bids are consecutive, and b2 has a lower utility than b1, from A's perspective. A similar property could be defined stating that an agent receives a global concession from another agent.
agent_consecutively_bids_to(γ, A, t1, b1, t2, b2, B) &
∀vA1, vA2 : real :
util(A, b1, vA1) & util(A, b2, vA2) ⇒
vA1 > vA2

configuration_differs(b1:BID, b2:BID)
Two bids b1 and b2 differ in configuration iff there is an issue that has a different value in both bids. Similar properties could be defined stating that two bids differ in configuration in at least x issues.
∃a: ISSUE, ∃v1, v2: VALUE:
value_of(b1, a, v1) &
value_of(b2, a, v2) &
v1 ≠ v2

agent_views_agent_makes_config_variation(γ:trace, A:AGENT, B:AGENT, t1:time, b1:BID, t2:time, b2:BID)

In the view of agent A, agent B varies the configuration, but not the utility. Note that one agent can both be agent A and B, or A and B can refer to different agents.

agent_consecutively_bids_to(γ, A, t1, b1, t2, b2, B) &
configuration_differs(b1, b2) &
∀vA1, vA2 : real :
util(A, b1, vA1) & util(A, b2, vA2) ⇒
vA1 = vA2

agent_views_agent_makes_strict_ε-progression(γ:trace, A:AGENT, B:AGENT, t1:time, b1:BID, t2:time, b2:BID, ε:real)

In the view of agent A, the two consecutive bids b1 and b2 made at times t1 and t2 by agent B show minimum ε-progression in utility iff the second bid is at least ε higher than the first bid. Note that one agent can both be agent A and B, or A and B can refer to different agents.

agent_consecutively_bids_to(γ, A, t1, b1, t2, b2, B) &
∀vA1, vA2 : real :
util(A, b1, vA1) & util(A, b2, vA2) ⇒
vA2 - vA1 > ε

strict_pareto_monotony(γ:trace, tb:time, te:time)

A negotiation process γ is Strictly Pareto-monotonous for the interval [t1, t2] iff for all subsequent bids b1, b2 in the interval b2 dominates b1:

∀t1, t2, ∀A, B: AGENT, ∀b1, b2: BID
[tb ≤ t1 < t2 ≤ te & is_followed_by(γ, A, t1, b1, B, t2, b2)]
⇒ strictly_dominates(γ, b2, b1, A, B)

weak_pareto_monotony(γ:trace, tb:time, te:time)

A negotiation process γ is Weakly Pareto-monotonous for the interval [t1, t2] iff for all subsequent bids b1, b2 in the interval b2 weakly dominates b1:

∀t1, t2, ∀A, B: AGENT, ∀b1, b2: BID
[tb ≤ t1 < t2 ≤ te & is_followed_by(γ, A, t1, b1, B, t2, b2)]
⇒ weakly_dominates(γ, b2, b1, A, B)

PART III NEGOTIATION

CHAPTER 9

Human vs. Computer Behaviour in Multi-Issue Negotiation

This chapter will appear as Bosse, T. and Jonker, C.M. (2005). Human vs. Computer Behaviour in Multi-Issue Negotiation. In: Ito, T., Hattori, H., Matsuo, T., and Zhang, M. (eds.), *Proceedings of the First International Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems, RRS'05*, pp. 10-25.

Human vs. Computer Behaviour in Multi-Issue Negotiation

Tibor Bosse[1] and Catholijn M. Jonker[2]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, NL-1081 HV Amsterdam, The Netherlands
tbosse@cs.vu.nl, http://www.cs.vu.nl/~tbosse
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl, http://www.nici.ru.nl/~catholj

Abstract. This paper presents two experiments that contribute to the comparison of human- versus computer behaviour in the domain of multi-issue negotiation. The experiments are part of an ongoing endeavour of improving the quality of computer negotiators when negotiating against human negotiators. The validity of the experiments was tested in a case study of closed multi-issue negotiation involving the ABMP negotiation software agents. The results indeed reveal a number of strengths and weaknesses of the ABMP agents. For example, the fairness of deals in negotiations performed purely by ABMP agents is better than the fairness of deals in the comparable negotiations in which humans were involved. Furthermore, in mixed negotiations (i.e., involving human- and software agents) the humans outperform the software agent with respect to the individual performance. Based on the results of the experiments, several suggestions are made to improve the ABMP agent's performance.

1. Introduction

Negotiation is an integral part of life, appearing, e.g., at the personal level of individual humans, at the level of companies, and at the level of countries. Over the years there is a steadily increasing stream of papers on the design of software agents for negotiation (see, e.g., [2]). These artificial negotiators are expected to be able to negotiate against other artificial- and against human negotiators. However, negotiation will never be delegated to artificial negotiators (also called agents), if their performance is not at least as good as that of human negotiators.

The need to establish the quality of negotiators implies a need for evaluation tools and experimental setups in which negotiators can be tested against each other. Note that in this formulation, negotiators can be either human or artificial. The SAMIN system for support and analysis of multi-issue negotiation [3] is a software environment that allows negotiators to play against other negotiators and that contains tools to evaluate the negotiation traces against a library of dynamic properties. SAMIN, although still under development, so far is the only such environment.

With respect to experimental research involving negotiation, most literature describes the result of testing different artificial agents against each other (e.g., [5]). Two welcome exceptions are [4] and [10]. In [4], a series of experiments is described where human participants interact with software agents in a Continuous Double Auction. In [10], the performance of a negotiating agent is compared with human performance in the game of Diplomacy. In both papers, the agents are found to outperform the humans. However, the authors of the second paper mention that this might be due to the fact that their participants were not always fully motivated. Moreover, the outcome of both papers is not

automatically transferable to other agents and domains. Every new agent will have to prove its worth in a new experiment. The experiment should test the agent with a covering spread of different profiles against human beings also using a covering spread of different profiles. Moreover, note that different domains place different demands on the negotiators. Another contribution to a more rigorous testing of (artificial) negotiation agents is the tournament of [7] in which several agents with different strategies were pitted against each other.

This paper therefore pleads for the development of a benchmark for negotiation that takes into account the different types of negotiation. For an overview of different types of negotiation, see [13]. For every type of negotiation, and for every form of additional constraints (e.g., regarding protocol, time limits and round limits) this benchmark should contain a set of domains with accompanying sets of profiles, a library of properties that should be used to evaluate the negotiations, and a set of experimental setups that should be performed when a new agent is introduced.

Having a standard for testing negotiation agents against humans is especially important. First of all, if the agents are not tested under the same conditions, the results of the tests cannot be compared. Secondly, experiments with humans are time and cost intensive. In our experience, explaining negotiation and the experimental settings to humans approximately takes one hour. On average a negotiation of human against agent takes roughly 30 minutes, a negotiation of human against human takes roughly 60 minutes. Furthermore, the group of humans involved should be sufficiently big to obtain statistically significant results.

If all agents are tested against humans under the same conditions, then the results of those tests can be compared, leading to a comparative analysis of the strengths and weaknesses of the agents involved.

This paper takes a step towards the proposed benchmark by providing the setup of two types of experiments for closed multi-issue negotiation (see, e.g. [13]) and showing their appropriateness by performing them in a case study. The first experiment concerns human-human negotiations, the second concerns human-computer negotiations. The first experiment needs only to be performed once for every domain included in the benchmark. The second needs to be performed for every domain and for every new agent.

The case study concerns a closed multi-issue negotiation on a number of attributes of second hand cars, using the ABMP agent introduced in [8] and the SAMIN system (of [3]) to carry out and analyse the negotiations.

Section 2 describes the formalisation of negotiation process dynamics in terms of negotiation states and traces. In addition, it shows how formal dynamic properties can be specified. The library of dynamic properties that are relevant for closed multi-issue negotiation is presented in Section 3. Section 4 explains briefly how the SAMIN system can use such properties for the analysis of negotiation traces. Next, Section 5 describes the experiments and case study. The results of these experiments for the case study are presented in Section 6. Section 7 completes the paper with a discussion and a description of future research plans.

2. Formalising Negotiation Processes

Negotiation is essentially a dynamic process. To analyse those dynamics, it is, therefore, relevant to formalise and study dynamic properties of such processes. For example, how does a bid at a certain point in time relate to bids at previous time points? The formalisation introduced in this section is based on the notions of negotiation process state and negotiation trace, which are introduced in Section 2.1 and 2.2. Based on these

concepts, it is demonstrated in Section 2.3 how formal dynamic properties can be specified.

2.1. Formalising States of a Negotiation Process

The *state* of a (one-to-one) negotiation process at a certain time point can be described as a combined state consisting of two states for each of the negotiating agents: S = < S1, S2 >, where S1 refers to the state of agent A, and S2 to the state of agent B. Each of these states include,

- the agent's own most recent bid
- its evaluation of its own most recent bid
- its evaluation of the other agent's most recent bid
- the history of bids from both sides and evaluations

To describe negotiation states a state ontology Ont is used. Example elements of this ontology are a sort BID for bids, and relations such as util(A, b, v) expressing that A's overall evaluation of bid b is v. Based on this ontology the set of ground atoms At(Ont) can be defined. A state is formalised by a truth assignment: At(Ont) $\rightarrow$ {t, f} to this set of ground atoms. The set of all states described by this ontology is denoted by States(Ont).

2.2. Negotiation Traces

A particular negotiation process shows a sequence of transitions from one state S from States(Ont) to another (next) state S' from States(Ont). A transition S $\rightarrow$ S' from a state S to S' can be classified according to which agents are involved. During such a transition each of the main state components (S1, S2) of the overall state S may change.

Negotiation traces are time-indexed sequences of negotiation states, where each successive pair of states is a negotiation transition. To describe such sequences a fixed *time frame* T is assumed which is linearly ordered. A *trace* γ over a state ontology Ont and time frame T is a mapping γ: T $\rightarrow$ STATES(Ont), i.e., a sequence of states γ_t (t $\in$ T) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering.

2.3. Dynamic Properties

To formally specify dynamic properties that express characteristics of dynamic processes (such as negotiation) from a temporal perspective an expressive language is needed. To this end the *Temporal Trace Language* TTL is used as a tool; cf. [9], which is briefly defined as follows.

The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, a certain state of the agent A during a negotiation process at time point t is indicated by state(γ, t, A). These state indicators can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t, A) |= p denotes that state property p holds in trace γ at time t in the state of agent A. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic

with sorts T for time points, Traces for traces and F for state formulae, using quantifiers and the usual first-order logical connectives such as ¬, ∧, ∨, ⇒, ∀, ∃.

As an example, consider the idea of making concession steps, which is a necessary action to take in order to move towards agreement (see, e.g., [5], [12]). A concession step can be expressed by the following dynamic property: "in trace γ, agent A makes a concession step between time points t1 and t2, if it makes bid b1 at t1 and it makes its next bid b2 at t2, and for agent A bid b2 has a lower utility than bid b1, while for the other agent B it has a higher utility". In TTL, this property can be formulated as follows:

```
concession_step(γ:TRACE, t1:time, t2:time, A:AGENT, B:AGENT) ≡
∃b1, b2:BID
   state(γ, t1, A) |= to_be_communicated_to_by(b1, B, A) &
   state(γ, t2, A) |= to_be_communicated_to_by(b2, B, A) &
   agent_consecutively_bids_to(γ, A, t1, b1, B, t2, b2) &
   ∀vA1, vA2, vB1, vB2: real :
      util(A,b1,vA1) & util(A,b2,vA2) & util(B,b1,vB1) & util(B,b2,vB2)
         ⇒ vA1 > vA2  &  vB1 < vB2
```

where util(A, b1, vA1) expresses that the utility of negotiator A with respect to bid b1 is the value vA1. See [13] for a definition of utility in negotiations. In the rest of the paper the formalisations will be kept to a minimum.

3. Properties of Negotiation Processes

To analyse the differences between human and computer negotiations, two categories of properties of the negotiations are investigated: those that concern the negotiators' *performance* in the negotiation, and those that concern the *steps* in their bidding behaviour. All properties are part of the current library for analysing negotiations. The library contains more properties, omitted here for reasons of space, see [3]. To improve readability, all properties are provided in an informal (natural language) notation, instead of the formal (TTL) notation introduced above.

3.1. Performance Properties

To measure the performance of the different parties in the negotiation, a number of different properties from the literature (e.g., [13], [14]) are included in the library:

- *Negotiator Final Utility*: a number between 0 and 1, indicating the negotiator's utility for the final bid in the negotiation (i.e., the bid that both parties agreed upon). The higher the utility, the higher the satisfaction of the negotiator. A high (but < 1) number does not mean that the negotiator could not have performed better. Furthermore, the utility of the one does not give any information about the utility of the other.
- *Pareto Distance*: a number between 0 and √2, indicating the shortest distance from the final bid in the negotiation to the *Pareto Efficient Frontier*. The Pareto Efficient Frontier (PEF) is the set of bids for which there exists no better bid for both parties, see e.g., [13]. Let x_i (respectively y_i) be the utility of negotiator X (respectively Y) for bid b_i. The distance d(b1, b2) between bids b1 and b2 is defined $\sqrt{((x1-y1)^2+(x2-y2)^2)}$. Thus, a short Pareto Distance means that there was little room for the negotiators to improve the outcome for both parties. However, this does not give much information about the fairness of the outcome.

- *Nash Distance*: a number between 0 and √2, indicating the distance from the final bid in the negotiation to the *Nash Point* (i.e., the point for which the product of both parties' utilities is maximal, see e.g., [13]). Since the Nash Point lies on the Pareto Efficient Frontier, a short Nash distance implies a short Pareto distance. The Nash Point is considered a fair outcome.
- *EPP Distance*: a number between 0 and √2, indicating the distance from the final bid in the negotiation to the *Equal Proportion of Potential Point* (also called the Kalai-Smorodinski Point, i.e., the point for which the difference between both parties' utilities is minimal, see e.g., [13]). Since the EPP Point also lies on the Pareto Efficient Frontier, a short EPP distance implies a short Pareto distance as well. The EPP Point is also considered a fair outcome.
- *Number of rounds*: a natural number, indicating the number of rounds the negotiation process took. One round consists of a bid made by the seller, followed by a bid made by the buyer. The smaller this number, the quicker an agreement was reached.

3.2. Step Properties

Besides observing the quality of the outcome, the trajectory of bids offered by each of the negotiators is of interest. Each trajectory is composed of the steps made by the negotiator. Every step satisfies exactly one of the following properties that are inspired by [3]:

- *Fortunate steps*: the next bid is better for yourself and better for the other agent
- *Concession steps*: the next bid is worse for yourself and better for the other agent
- *Selfish steps*: the next bid is better for yourself and worse for the other agent
- *Unfortunate steps*: the next bid is worse for yourself and worse for the other agent

As argued in [6], agreement in multi-issue negotiation can often be reached quicker if both parties make concessions. In the experiments described in the next section, it will be investigated to what extent the different types of steps are used, both by human and computer negotiators.

4. SAMIN: The System for Analysis of Multi-Issue Negotiation

To carry out and analyse a number of experiments in negotiation (see next section), the SAMIN system by [3] was used. This Section briefly explains the working of the system.

At the top level, SAMIN consists of three components: an *Acquisition Component*, an *Analysis Component* and a *Presentation Component*, see Figure 1 Here, the solid arrows indicate data flow. The dotted arrows indicate that each component can be controlled separately by the user. The Acquisition Component is used to acquire the input necessary for analysis. The Analysis Component is used to perform the actual analysis (i.e., checking which properties hold for the negotiation process under analysis). Finally, the Presentation Component is used to present the results of the analysis in a user-friendly format. Furthermore, SAMIN maintains a library of properties, templates of properties, bid ontologies, and profile ontologies (not shown in Figure 1. The working of the three components will be described briefly in the next subsections. For more details, see [3].

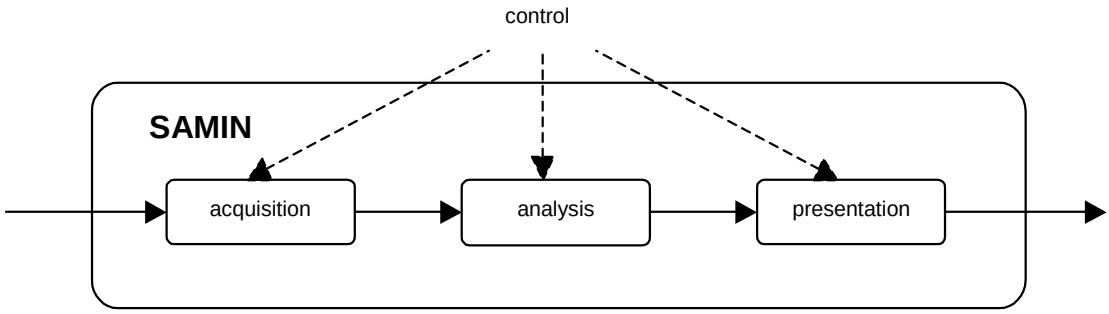


Figure 1. Global overview of the SAMIN architecture

4.1. The Acquisition Component

The acquisition component is used to obtain the required input for the analysis. It consists of an *ontology editor*, a *dynamic property editor* and a *trace determinator*.

The ontology editor is used for the construction of bid ontologies and profile ontologies necessary to automatically interpret the bids exchanged by the negotiators, and to automatically interpret the profiles of the negotiators. The ontology editor is typically used to construct a bid ontology and a profile ontology, thus allowing the user to identify the issues to be negotiated, the values that each of these issues can take, and the structure of bids, in the bid ontology. A *profile* is a description of the preferences of the negotiator within the particular negotiation domain. Thus, in specifying the profile ontology the user identifies the possible evaluations that can be given to values, and the utility functions of bids.

The dynamic property editor supports the gradual formalisation of dynamic properties in TTL format. The editor offers a user interface that allows the analyst to construct dynamic properties, represented in a tree-like format.

The trace determinator can be used interactively with the analyst to determine what traces to use in the analysis. The user can interactively locate the files containing the traces to be checked. The traces themselves can be of three categories: (human) empirical traces, simulated traces, and mixed traces. An empirical trace is the result of an existing human negotiation process. A simulated trace is the result of an automated negotiation process. A mixed trace is the result of a human negotiating with a software agent. To support the acquisition of traces of all three types, a dedicated interface has been created for SAMIN.

4.2. The Analysis Component

The analysis component currently consists of a *logical analyser* that is capable of checking properties against traces. To this end, the tool takes a dynamic property in TTL format and one or more traces as input, and checks whether the dynamic property holds for the traces.

Traces are represented by sets of Prolog facts of the form `holds(state(m1, t(2)), a, true)` where `m1` is the trace name, `t(2)` time point 2, and `a` is a state property as introduced in Section 2.1. The above example indicates that state formula `a` is true in trace `m1` at time point `2`. The Analysis Component basically uses Prolog rules for the predicate `sat` that reduce the satisfaction of the temporal formula finally to the satisfaction of atomic state formulae at certain time points, which can be read from the trace representation. Examples of such reduction rules are:

```
sat(and(F,G)) :- sat(F), sat(G).
sat(not(and(F,G))) :- sat(or(not(F), not(G))).
sat(or(F,G)) :- sat(F).
sat(or(F,G)) :- sat(G).
sat(not(or(F,G))) :- sat(and(not(F), not(G))).
```

In addition, if a dynamic property does not hold in a trace, then the software reports the places in the trace where the property failed.

4.3. The Presentation Component

The presentation component currently includes a tool that visualises the negotiation space in terms of the utilities of both negotiators. This *visualisation tool* plots the bid trajectory in a 2-dimensional plane, see Figure 2. The utilities are real values that indicate how a particular bid is evaluated by a negotiator.

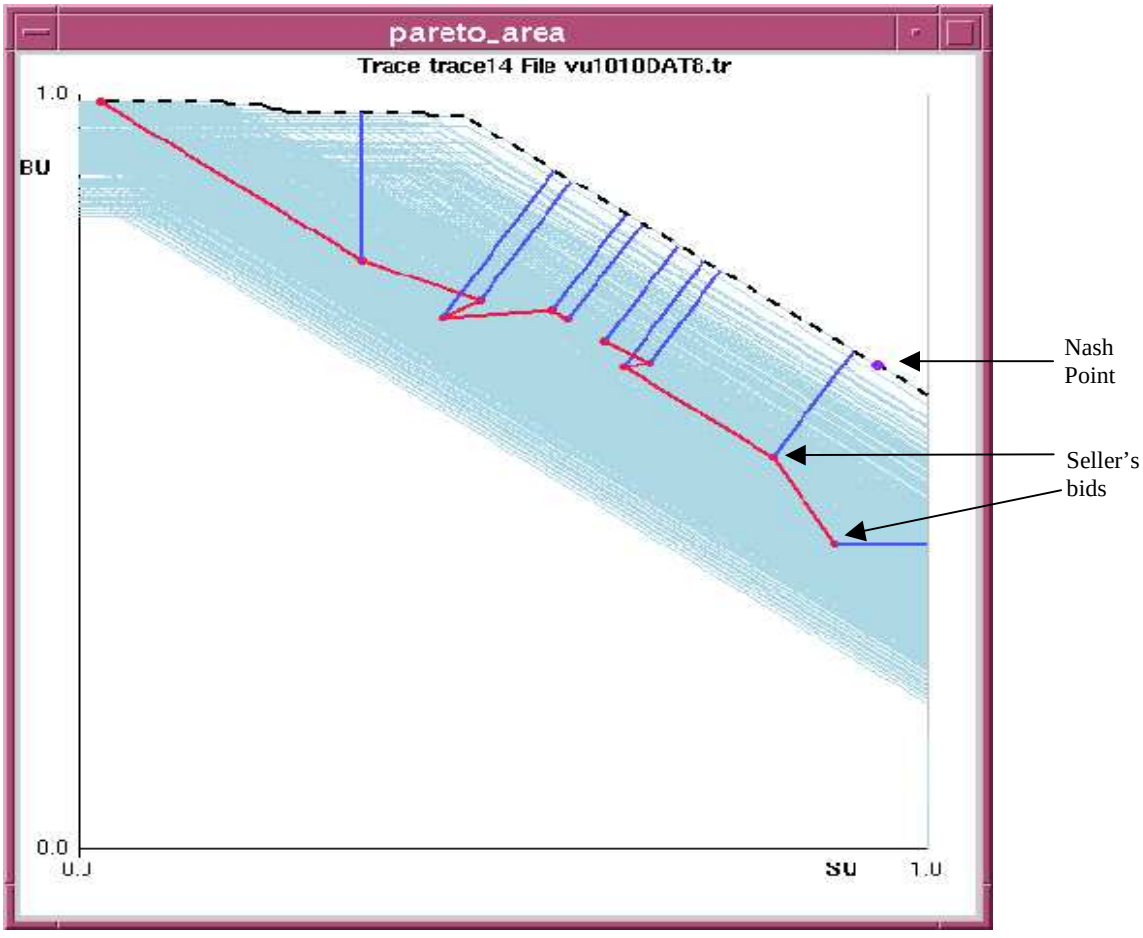


Figure 2. SAMIN Visualisation Tool

In Figure 2, the seller's utility of a bid is on the horizontal axis, and the buyer's utility is on the vertical axis. The light area corresponds to the space of possible bids. In this area, each curve is a continuous line, corresponding to a different combination of discrete issues. The specific position on the line is determined by the continuous issue 'price'. Since in this particular domain 4 discrete issues with 5 possible values occur (see next section), there are already 625 (= 5^4) different curves. In this figure, the sequences of actual bids made by both buyer (left) and seller (right) are indicated by the dark points that are connected by the two angular lines. The upper-left point indicates the buyer's first

bid, and the lower-right point indicates the seller's first bid. The dotted line indicates the Pareto Efficient Frontier according to the profiles of the negotiating agents, and the short dark lines show the distance from each bid to this frontier. The small dot that is plotted on the Pareto Efficient Frontier corresponds to the Nash Point. From this picture, it is clear that both negotiators make more and more concessions (their bids converge towards each other). Eventually, they reach a point that does not lie on the Pareto Efficient Frontier, but is rather close to it anyhow.

5. The Experiments

Pre-experiments with 10 participants showed that a Human-Human (HH) negotiation for multi-issue negotiations for non-trivial domains takes approximately one hour to complete. The subjects showed signs of fatigue, and when asked to perform another negotiation, they showed a lack of motivation. The negotiations were performed much quicker, but the results were obviously sub-optimal. On the basis of these observations, the experiments proposed in this paper are limited to one negotiation per human.

Sections 5.1 and 5.2 describe the setup of the experiments. The case study, a negotiation about second hand cars involving the ABMP agents and the SAMIN system, is presented in Section 5.3. The results of the analysis of the acquired traces are presented in Section 6. The strategy used by the ABMP agent can be summarised by the following steps (see [8] for details):

1. For each negotiation round, determine evaluations of the attributes of the previous bids.
2. Aggregate these evaluations into overall utilities of these previous bids.
3. Determine which concession step will be made for the next bid, expressed in terms of the overall utility; this provides a target utility.
4. To obtain the next bid, given the target utility, determine target attribute evaluation values, according to some distribution over attributes (chosen in such a manner that they aggregate exactly to the target utility)
5. For each of these target attribute evaluation values, choose an attribute value that has an evaluation value as close as possible to the target evaluation value for the attribute.

Since this strategy is based on the idea of monotonic concession, we hypothesized that the computer negotiator mainly uses concession steps. On the contrary, human negotiators probably are more diverse in their behaviour.

5.1. Experiment 1: HH

Participants. Gather a selection of humans representative of the adult population. The selection should contain enough humans to possibly gain statistically significant results. The size of the group depends on the number of variables in the domain.

For the case study eighteen subjects participated in this experiment. The make up of the group was not representative of the adult population in general, but was representative of the population of AI students in The Netherlands. The group consisted of 12 males and 6 females. All participants were students in AI, their age varying between 19 and 27 years.

Preparation. Before starting the experiment, the participants are to be provided enough background information to be able to perform the negotiation and use the software environment used to register the negotiations. The participants should be motivated to do their best during the negotiation.

In the case study the participants were motivated by the challenge to obtain a high utility, and to perform better than the computer in the corresponding Computer-Computer negotiation process (CC) they were also allowed to perform. The participants formed 9 groups of two persons, and each group was assigned to a computer.

Method. Each group has to participate in two negotiation processes: a HH process and a CC process. In the HH process, one person is assigned the role of the buyer, and the other one is assigned the role of the seller. The human buyer negotiates with the human seller (both using their own profile). The subjects are not allowed to look at the screen while the other party makes a bid. In the CC process, a computer buyer negotiates with a computer seller (both using the profile of the corresponding human negotiator). By keeping the negotiation profile stable over the two processes, it is guaranteed that the utility spaces remains the same, and that the resulting traces are thus comparable.

5.2. Experiment 2: HC

Participants. Gather a selection of humans representative of the adult population. The selection should contain enough humans to possibly gain statistically significant results.

In the case study 76 subjects (43 males and 33 females) participated in this experiment. The experiment took place during an introductory course for family members of AI students. Most of the participants (about 75%) were parents of the students, their age varying between 45 and 55 years. The other 25% were brothers and sisters of the students, their age varying between 17 and 24 years. Almost all of the participants did not have any background in AI. Education and occupation were a fair representation of the general population in The Netherlands.

Preparation. Before starting the experiment, the participants are to be provided enough background information to be able to perform the negotiation and use the software environment used to register the negotiations.

In the case study the game theoretic notions were treated a bit less thoroughly than in the case study for Experiment 1. The participants formed 38 teams of two persons, and each team was assigned to a computer. Each team was told that they could negotiate as a team against the computer. This deviation was necessary for that occasion, due to a lack of available computers.

Method. Each group has to participate in two negotiation processes: a Human-Computer (HC) process and a CC process. In the HC process, all teams play the same role (e.g., the buyer role), and use their own personal profile. In the CC process, a computer buyer uses the profile of the human team. By keeping the negotiation profile stable over the two processes, it is guaranteed that the utility spaces remain the same, and that the resulting traces are thus comparable.

5.3. Case study

The case study concerned a multi-issue closed negotiation on second hand cars. To support and analyse the experiments, the SAMIN system [3] was used.

The object of negotiation in the case study is a particular second hand car, for which the relevant issues are cd_player, extra_speakers, airco, drawing_hook and price.

Consequently, a bid consists of an indication of which CD player is meant, which extra speakers, airco and drawing hook, and what the price of the bid is. The goal of the negotiators is to find agreement upon the values of the four accessories and the price. Here, the price issue has a continuous value, whilst the other four issues have a value from discrete sets.

Before the negotiation starts, both parties specify their *negotiation profile*, see [8]. SAMIN offers negotiators a graphical interface to specify their personal profile. In addition to this negotiation profile, the seller is also provided with a *financial profile*, describing for each issue how much it costs, both to buy it and to build it into the car. Since we focus on closed negotiation, none of the profiles will be available for the other negotiator. However, SAMIN has access to both profiles.

During the negotiation, all subjects could input their bids within a special interface that also shows the history of bids. To help human negotiators generating their bids, the system offers a special tool that allows the player to calculate the utility of a bid before passing it to the other party. The resulting negotiation traces were logged by the system, so that they could be used for the purpose of analysis.

6. Results

Using the SAMIN system, the properties for multi-issue negotiation introduced in Section 3 have been automatically checked against the traces that resulted from the experiments. This section shows the results of the analysis. Section 6.1 focuses on Experiment 1, and Section 6.2 focuses on Experiment 2. Each section distinguishes between the properties concerning the parties' performance, and those concerning their bidding behaviour. Section 6.3 discusses the most important results of both experiments.

6.1. Experiment 1

6.1.1. Performance Properties

The results with respect to the performance of the negotiators in Experiment 1 are shown in Table 1. The first row contains the mean outcomes over all 9 HH traces. The second row contains the mean outcomes over all 9 CC traces. To test whether the differences between these two means were significant, paired t-tests have been performed, of which the results are shown in the last two rows. For example, the first column states that in the HH traces, the mean utility of the buyer was 0.87, that in the CC traces, the mean utility of the buyer was 0.88, but that this difference was not significant ($t=0.38$, $p<0.717$).

	Buyer Utility	Seller Utility	Pareto Distance	Nash Distance	EPP Distance	Number of rounds
HH traces	0.87	0.80	0.05	0.22	0.16	7.00
CC traces	0.88	0.89	0.03	0.12	0.06	8.00
t-value	0.376	2.807	-0.786	-3.988	-3.463	1.540
p-value	0.717	0.023	0.455	0.004	0.009	0.146

Table 1. Performance in Experiment 1

As can be seen in the table, the results in the second, fourth and fifth column are significant. Thus, the following conclusions can safely be drawn from the experiments:

- the seller's mean utility was significantly higher in the CC traces than in the HH traces ($t=2.81$, $p<0.023$)

- the mean Nash Distance was significantly shorter in the CC traces than in the HH traces ($t=-3.99$, $p<0.004$)
- the mean EPP Distance was significantly shorter in the CC traces than in the HH traces ($t=-3.46$, $p<0.009$)
- with respect to the other properties, there was no significant difference between the HH traces and the CC traces

6.1.2. Step Properties

The results with respect to the step properties in Experiment 1 are shown in Table 2.

	Fortunate (S+ O+)	Concession (S- O+)	Selfish (S+ O-)	Unfortunate (S- O-)
HH, buyer	3 (6.52%)	36 (78.26%)	2 (4.35%)	5 (10.87%)
HH, seller	5 (11.36%)	32 (72.73%)	4 (9.09%)	3 (6.82%)
CC, buyer	0 (0%)	58 (89.23%)	0 (0 %)	7 (10.77 %)
CC, seller	0 (0 %)	48 (82.76%)	0 (0 %)	10 (17.24 %)

Table 2. Steps made in Experiment 1

The numbers between brackets are the percentages with respect to the total amounts of steps. The first row shows the steps made by the (human) buyers in the HH traces. The second row shows the steps made by the (human) sellers in the HH traces. Similarly, the third and fourth row show the steps made by the computer buyers, respectively sellers in the CC traces. For example, the first cell indicates that in the HH traces, all (human) buyers together made 3 fortunate steps, which is 6.52% of the total amount of steps they made.

This table clearly shows that both the human and computer negotiators primarily made concession steps. Besides that, the computers made some unfortunate steps, more than the humans did. The humans were more diverse in their behaviour, since they also made some selfish and some fortunate steps.

6.2. Experiment 2

6.2.1. Performance Properties

The results with respect to the performance of the negotiators in Experiment 2 are shown in Table 3. The first row indicates the mean outcomes over all 38 HC traces, the second row indicates the mean outcomes over all 38 CC traces, and the last two rows show the results of the paired t-tests.

	Buyer Utility	Seller Utility	Pareto Distance	Nash Distance	EPP Distance	Number of rounds
HC traces	0.89	0.72	0.05	0.30	0.23	8.84
CC traces	0.87	0.83	0.06	0.17	0.10	8.91
t-value	-1.729	3.684	0.309	-5.161	-6.228	0.066
p-value	0.092	0.001	0.759	0.000	0.000	0.948

Table 3. Performance in Experiment 2

The results in the second, fourth and fifth column are significant. Thus, the following conclusions can safely be drawn from the experiments:

- the seller's mean utility was significantly higher in the CC traces than in the HC traces ($t=3.68$, $p<0.001$)
- the mean Nash Distance was significantly shorter in the CC traces than in the HC traces ($t=-5.16$, $p<0.000$)
- the mean EPP Distance was significantly shorter in the CC traces than in the HC traces ($t=-6.23$, $p<0.000$)
- with respect to the other properties, there was no significant difference between the HC traces and the CC traces

6.2.2. Step Properties

The results with respect to the step properties in Experiment 2 are shown in Table 4. This table shows the same trends as Table 2. Again, both the human and computer negotiators primarily made concession steps. Besides that, the computers made some fortunate and some unfortunate steps, and no selfish steps. The humans, on the other hand, were more diverse in their behaviour. They made significantly more steps in the non-concession categories than the computer.

	Fortunate (S+ O+)	Concession (S- O+)	Selfish (S+ O-)	Unfortunate (S- O-)
HC, buyer	23 (7.62%)	232 (76.82%)	17 (5.63%)	30 (9.93%)
HC, seller	2 (0.68%)	251 (85.37%)	0 (0 %)	41 (13.95%)
CC, buyer	0 (0 %)	287 (94.41 %)	0 (0 %)	17 (5.59 %)
CC, seller	0 (0 %)	267 (90.51 %)	0 (0 %)	28 (9.49 %)

Table 4. Steps made in Experiment 2

6.3. Discussion

Since the profiles used in Experiment 1 differ from those in Experiment 2, it makes no sense to compare the exact data from both experiments with each other. To illustrate this, suppose that in Experiment 1 the profiles of buyer and seller were generally much more similar than in Experiment 2. In that case, in Experiment 1 it would be much easier for both parties to obtain high utility values. Nevertheless, it is possible to compare the general trends one can observe in both experiments.

One trend observed in both experiments, is that the Nash distance and the EPP distance (both measures for fairness of the negotiation) were very short in the CC traces. Table 1 shows that these distances were significantly shorter in the CC traces than in the HH traces, and Table 3 shows that they were shorter in the CC traces than in the HC traces. Furthermore, these distances seem to be shorter in the HH traces than in the HC traces. Thus, the CC negotiations turned out to have the "fairest" outcome, followed by the HH traces. The outcomes of the HC traces were the least balanced. This can be seen in the first two cells in Table 3, where the mean (human) buyer utility (0.89) was much higher than the mean (computer) seller utility (0.72). This is an important finding, because when the same negotiation spaces are explored by two computer negotiators, the buyer utility

hardly drops (0.87), whilst the seller utility increases significantly (0.83). Apparently the computer seller is not robust to being exploited by a human buyer. This observation is supported by the data in Table 2 and 4. In both situations, the computers made more unfortunate steps than the humans. In addition, the computer sellers made more unfortunate steps than the computer buyers. A detailed analysis of the traces revealed that the computer seller mostly makes these unfortunate steps when it raises the price of the bid in order to compensate for a better component. Better results might be obtained by making this price compensation a bit lower.

Another explanation of the fact that CC negotiations seem to reach fairer outcomes than HH and HC traces might come from the similarity of the ABMP buyer and seller agents. Both agents follow the same concession strategy, and the parameters that can be used to tune and vary the behaviour of the ABMP agents, were the same in both agents. This explanation can be tested by running the CC negotiations again with different parameter settings for both agents.

A last important finding concerns the diverse bidding behaviour by humans. As shown in Table 2, human negotiators sometimes make steps that improve the utility for both parties. Of course, doing this has the risk of making selfish steps. In its current state, the ABMP agent hardly makes these kinds of steps. Nevertheless, in some cases the unpredictable human behaviour actually resulted in better results. Therefore, it could be beneficial to incorporate these strategies in the software agent as well. Furthermore, it might even help to introduce some (seemingly) unfortunate steps. In many CC traces, the parties found agreement upon the "discrete" issues very quickly, leaving only the price to negotiate upon. However, introducing more unfortunate steps (and thereby changing the values of the discrete issues) might help to escape from "local optima", in the same manner as simulated annealing techniques in evolutionary computing do. A similar idea to improve performance is to consider so-called *weak* concessions: the negotiator changes some values while keeping his utility stable.

7. Conclusion and Future Work

This paper pleads for a benchmark for negotiations. The benchmark is argued to be essential for the improvement of artificial negotiators. The benchmark should provide a general framework and software environment for the comparison of the negotiators with human and artificial negotiators. The work of [3], [4], [7], and [10] can be seen as earlier contributions to such a benchmark.

The two experimental setups presented in this paper were shown to contribute to that benchmark as well. The experiments focus on the comparison of human with computer behaviour in the domain of one-to-one multi-issue negotiation. To validate the use of the experimental setups, a case study was performed. In the first experiment, human-human negotiation was compared with computer-computer negotiation. In the second experiment, human-computer negotiation was compared with computer-computer negotiation. Both experiments yielded a number of interesting results, which can be used to improve the quality of software negotiators in the future.

In fact, these results show that at a number of points the agent of the case study already outperforms the humans. These points mainly involve the fairness of the negotiating outcome. However, with respect to the individual performance (i.e., without caring about the utility of the other party) there is still some room for improvement. Based on the results of the experiments, some suggestions have been made to improve the agent of the case study. Examples are decreasing the price compensations by the seller, introducing more switches between issues, and introducing weak concessions.

With respect to related work, the existing literature describes several other systems that aim at the formal analysis of negotiation processes. Most of these systems focus on CC negotiation, not on HC, CH or HH negotiation. Nevertheless, a number of papers also mention dynamic properties of negotiation processes, e.g., [11], [14], [15]. Like in the current paper, a number of these properties are oriented towards the outcome of the negotiation, in terms of Pareto distance or Nash distance. In addition to this, our paper also identifies a number of properties that are geared towards the *dynamics* of the negotiation process instead of the outcome. In return, in [11], [14], [15] some additional properties are mentioned that are oriented towards rationality and use of resources.

For future research, it is planned to develop and perform more experiments. In particular, a series of experiments will be developed and performed where in all possible combinations (CC, HH, HC, CH) the same profiles are used. Here, especially the CH case (computer buyer vs. human seller) is interesting. Another plan is to extend the SAMIN framework to ease experiments in other domains and open the system for any artificial agent. From the current results, it is still debatable to what extent the conclusions drawn may be generalised. Although the particular strategies of the ABMP agents were designed to be representative for most agent-based strategies in multi-issue negotiation, more experiments with other software agents (or with ABMP agents using different parameter settings) would be welcome.

Finally, from a psychological perspective, it is planned to perform a "Turing test" for negotiation. Following the ideas of [1], a tournament may be set up involving a number of human and automated negotiators. In such a setting, the human participants will have to find out whether they are dealing with an automated or a human negotiation partner.

Acknowledgments

The authors are grateful to Lourens van der Meij for his contribution to the development of the software environment, and to all participants in the case study.

References

[1] Arifovic, J. (2005). *The Implementation of the Turing Tournament: A Report*. Simon Fraser University, Burnaby, Canada. Technical Report.

[2] Beam, C. and Segev, A. (1997). Automated negotiations: a survey of the state of the art. *Wirtschaftsinformatik*, 39(3), 1997, pp. 263–268.

[3] Bosse, T., Jonker, C.M., and Treur, J. (2004). Experiments in Human Multi-Issue Negotiation: Analysis and Support. In: Jennings, N.R., Sierra, C., Sonenberg, L., and Tambe, M. (eds.), *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'04*. IEEE Computer Society Press, 2004, pp. 672-679.

[4] Das, R., Hanson, J.E., Kephart, J.O., and Tesauro, G. (2001). Agent-Human Interactions in the Continuous Double Auction. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI'01*. Morgan Kaufman, 2001.

[5] Faratin, P., Sierra, C., and Jennings, N.R. (1998). Negotiation decision functions for autonomous agents. In: *International Journal of Robotics and Autonomous Systems*, vol. 24(3-4), 1998, pp. 159 -182.

[6] Gutman, R. and Maes, P. (1998). Agent-mediated Integrative Negotiation for Retail Electronic Commerce. In: P. Noriega and C. Sierra (eds.), *Agent Mediated Electronic Commerce*, Lecture Notes in AI, vol. 1571, 1998, pp. 70-90.

[7] Henderson, P., Walters, B., Crouch, S. and Ni, Q. (2003). A comparison of some negotiation algorithms, in Agent Technologies, Infrastructure, Tools and Applications for E-Services, Springer-Verlag, Lecture Notes in AI, vol. 2592, 2003, pp. 137-150.

[8] Jonker, C.M. and Treur, J. (2001). An Agent Architecture for Multi-Attribute Negotiation. In: B. Nebel (ed.), *Proceedings of the 17th International Joint Conference on AI, IJCAI'01*, 2001, pp. 1195 - 1201.
[9] Jonker, C.M. and Treur, J. (2002). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.
[10] Kraus, S. and Lehmann, D.J. (1995). Designing and building a negotiating automated agent. *Computational Intelligence*, 11(1), 1995, pp. 132-171.
[11] Lomuscio, A.R., Wooldridge, M., and Jennings. N.R. (2000). A classification scheme for negotiation in electronic commerce, In: *International Journal of Group Decision and Negotiation*, vol. 12(1), January 2003.
[12] Pruitt, D.G. (1981). *Negotiation Behavior*, Academic Press.
[13] Raiffa, H. (1996). *Lectures on Negotiation Analysis*, PON Books, Program on Negotiation at Harvard Law School, 513 Pound Hall, Harvard Law School, Cambridge, Mass. 02138, 1996.
[14] Rosenschein, J.S. and Zlotkin, G. (1994). Rules of Encounter: Designing Conventions for Automated Negotiation among Computers. The MIT Press, Cambridge, MA, 1994.
[15] Sandholm, T. (1999). Distributed rational decision making. In: Weiss, G., (ed.), *Multi-agent Systems: A Modern Introduction to Distributed Artificial Intelligence*, MIT Press, 1999, pp. 201 – 258.

PART IV ADAPTIVITY

CHAPTER 10

Formalisation and Analysis of the Temporal Dynamics of Conditioning

This chapter will appear as Bosse, T., Jonker, C.M., Los, S.A., Torre, L. van der, and Treur, J. (2005). Formalisation and Analysis of the Temporal Dynamics of Conditioning. *Cognitive Systems Research Journal*.

Part of this chapter appeared as Bosse, T., Jonker, C.M., Los, S.A., Torre, L. van der, and Treur, J. (2005). Formalisation and Analysis of the Temporal Dynamics of Conditioning. In: Mueller, J.P. and Zambonelli, F. (eds.), *Proceedings of the Sixth International Workshop on Agent-Oriented Software Engineering, AOSE'05*, pp. 157-168.

Formalisation and Analysis of the Temporal Dynamics of Conditioning

Tibor Bosse[1], Catholijn M. Jonker[1], Sander A. Los[2],
Leendert van der Torre[3], and Jan Treur[1,4]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, jonker, treur}@cs.vu.nl
URL: http://www.cs.vu.nl/~{tbosse, jonker, treur}
[2] Vrije Universiteit Amsterdam, Department of Cognitive Psychology,
Van der Boechorststraat 1, 1081 BT Amsterdam, The Netherlands
sa.los@psy.vu.nl
URL: http://www.cs.vu.nl/~cogsci/cogpsy/sander
[3] Centrum voor Wiskunde en Informatica,
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
torre@cwi.nl
URL: http://homepages.cwi.nl/~torre
[4] Universiteit Utrecht, Department of Philosophy,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Abstract. In the literature classical conditioning is usually described and analysed informally. If formalisation is used, this is often based on mathematical models based on difference or differential equations. This paper explores a formal description and analysis of a conditioning process based on logical specification and analysis methods of dynamic properties of the conditioning process. Specific types of dynamic properties are global dynamic properties, decribing properties of the process as a whole, or local dynamic properties, describing properties of basic steps in a conditioning process. If the latter type of properties are specified in an executable format, they provide a temporal declarative specification of a simulation model. By a software environment these local properties can be used to actually perform simulation. Global properties can be checked automatically for simulated or other traces. Using these methods the properties of conditioning processes informally expressed by Los and van den Heuvel (2001) have been formalised and verified against a specification of local properties based on Machado (1997)'s differential equation model.

1. Introduction

A common approach to describe dynamics of cognitive processes is by relating sensory, cognitive or behavioural states to previous or subsequent states. For example, this is shown in approaches from what in Philosophy of Mind is called the functionalist perspective, where the functional role of a mental state property is defined by its predecessor and successor states. Also in Dynamical Systems Theory (DST), a relatively new approach to describe the dynamics of cognitive processes, which subsumes connectionist modelling, e.g., (Port and Gelder, 1995), relations of a state with previous and subsequent states are central. Gelder and Port (1995) briefly explained what a dynamical system is in the following manner. A *system* is a set of changing aspects (or state properties) of the world. A *state* at a given point in time is the way these aspects or state properties are at that time; so a state is characterised by the state properties that hold.

The set of all possible states is the *state space*. A *behaviour* of the system is the change of these state properties over time, or, in other words, a succession or sequence of states within the state space. Such a sequence in the state space can be indexed, for example, by natural numbers (*discrete* case) or real numbers (*continuous* case), and is also called a *trace* or *trajectory*. Given these notions, the notion of *state-determined system*, adopted from (Ashby, 1960) is taken as the basis to describe what a dynamical system is:

> *'A system is state-determined only when its current state always determines a unique future behaviour. (..) the future behaviour cannot depend in any way on whatever states the system might have been in* before *the current state. In other words, past history is irrelevant (or at least, past history only makes a difference insofar as it has left an effect on the current state). (..) the fact that the current state determines future behaviour implies the existence of some* rule of evolution *describing the behaviour of the system as a function of its current state.' (Gelder and Port, 1995, p. 6).*

The assumption of state-based systems, which is fundamental for DST, entails a *local modelling perspective* where states are related to immediate predecessor and successor states. This local modelling perspective, which DST has in common with the functional perspective based on causal relations mentioned earlier, is especially useful for simulation purposes. In addition, it is often relevant to check whether certain processes show emergent properties at a global level. In order to check this automatically, these *global temporal relations* between states need to be formalised. However, this is beyond modelling approaches like DST.

In more sophisticated cognitive processes, a cognitive state or a behaviour can be better understood in a more global manner, e.g., in the way in which it depends on a longer history of experiences or inputs. In experimental research, examples of such phenomena are usually indicated as inter-trial or adaptive effects. Approaches to model such more sophisticated cognitive processes require means to express a more advanced *temporal complexity* than for the less sophisticated processes, where direct succession relations between states suffice as appropriate means.

Various types of adaptive or learning behaviour are known and have been studied in some depth. For example, for various forms of learned stimulus-response behaviours it has been studied how they are determined by an attained cognitive state of the organism. Still, the question remains how such attained cognitive states themselves depend on the previous history, for example on a particular training session extending over a longer time period. Often insight is obtained by formulating such more complex temporal relationships. However, usually such complex temporal relationships are expressed purely informally, due to the lack of modelling techniques that reach beyond the local perspective.

In order to temporally relate states to states at other points in time in a more global manner, modelling approaches to dynamical systems of a different type have recently been proposed. Within the areas of Computer Science and Artificial Intelligence techniques have been developed to analyse the dynamics of phenomena using logical means. Examples are dynamic and temporal logic, and event and situation calculus; e.g., (Eck, et al. 2001; Kowalski and Sergot, 1986; Reiter, 2001). These logical techniques allow to consider and relate states of a process at different points in time, and in this sense reach beyond the local perspective. The form in which these relations are expressed can cover qualitative aspects, but also quantitative aspects.

This paper addresses temporal aspects of conditioning and illustrates the usefulness of a logical approach for the analysis and formalisation of such processes both at a local and at a more global level. First a local perspective model for temporal conditioning in a high-level executable format is presented. This executable model can be compared to (and was

inspired by) Machado (1997)'s differential equation model. Some simulation traces are shown and compared to traces of Machado (1997)'s model.

Next, as part of a non-local perspective analysis, a number of relevant dynamic properties of the conditioning process are identified and formalised. These dynamic properties were obtained by formalising the informally expressed properties to characterise temporal conditioning processes, as put forward by Los and Van Den Heuvel (2001). It has been automatically verified that (under reasonable conditions) these global dynamic properties are satisfied by the simulation traces.

2. Temporal Dynamics of Conditioning

2.1. Basic Concepts of Conditioning

Research into conditioning is aimed at revealing the principles that govern associative learning. To this end, several experimental procedures have been developed. In classical conditioning, an organism is presented with an initially neutral conditioned stimulus (e.g., a bell) followed by an unconditioned stimulus (e.g., meat powder) that elicits an innate or learned unconditioned response in the organism (e.g. saliva production for a dog). After acquisition, the organism elicits an adaptive conditioned response (also saliva production in the example) when the conditioned stimulus is presented alone. In operant conditioning, the production of a certain operant response that is part to the volitional repertoire of an organism (e.g., bar pressing for a rat) is strengthened after repeated reinforcement (e.g., food presentation) contingent on the operant response.

In their review, Gallistel & Gibbon (2000) argued that these different forms of conditioning have a common foundation in the adaptive timing of the conditioned (or operant) response to the appearance of the unconditioned stimulus (or reinforcement). This feature is most apparent in an experimental procedure called trace conditioning, in which a blank interval (or 'trace') of a certain duration separates the conditioned and unconditioned stimulus (in classical conditioning) or subsequent reinforcement phases (in operant conditioning). In either case, the conditioned (or operant) response obtains its maximal strength, here called *peak level*, at a moment in time, called *peak time*, that closely corresponds to the moment the unconditioned stimulus (or reinforcement) occurs.

For present purposes, we adopt the terminology of an experimental procedure that is often used to study adaptive timing and the possible role of conditioning in humans. In this procedure, a trial starts with the presentation of a *warning stimulus* (S1; comparable to a conditioned stimulus). After a blank interval, called the *foreperiod* (FP), an *imperative stimulus* (S2, comparable to an unconditioned stimulus) is presented to which the participant responds as fast as possible. The *reaction time* (RT) to S2 is used as an estimate of the conditioned state of preparation at the moment S2 is presented.

In this type of research, FP is usually varied at several discrete levels. That is, S2 can be presented at several moments since the offset of S1, which are called *critical moments*. The moment that is used for the presentation of S2 on any given trial is called the *imperative moment* of that trial. A final distinction concerns the way the different levels of FP are presented to the participant. In a *pure block*, the same FP is used across all trials of that block (and varied between different pure blocks). That is, in a pure block there is one critical moment that corresponds to the imperative moment on each trial. In a *mixed block*, all levels of FP occur randomly across trials. That is, a mixed block has several critical moments, but on any specific trial, only one of the moments is the imperative moment.

2.2. Modelling Conditioning by Differential Equations

Machado (1997) presented a basic model of the dynamics of a conditioning process. The structure of this model, with an adjusted terminology as used by Los, Knol & Boers (2001), is shown in Figure 1. The model posits a layer of *timing nodes* (Machado calls these *behavioral states*) and a single *preparation node* (called *operant response* by Machado). Each timing node is connected both to the next timing node and to the preparation node. The connection between each timing node and the preparation node (called *associative link* both by Machado and within the current paper) has an adjustable weight associated to it. Upon the presentation of a warning stimulus, a cascade of activation propagates through the timing nodes according to a regular pattern. Owing to this regularity, the timing nodes can be likened to an internal clock or pacemaker. At any moment, each timing node contributes to the activation of the preparation node in accordance with its activation and its corresponding weight. The activation of the preparation node reflects the participant's preparatory state, and is as such related to reaction time for any given imperative moment.

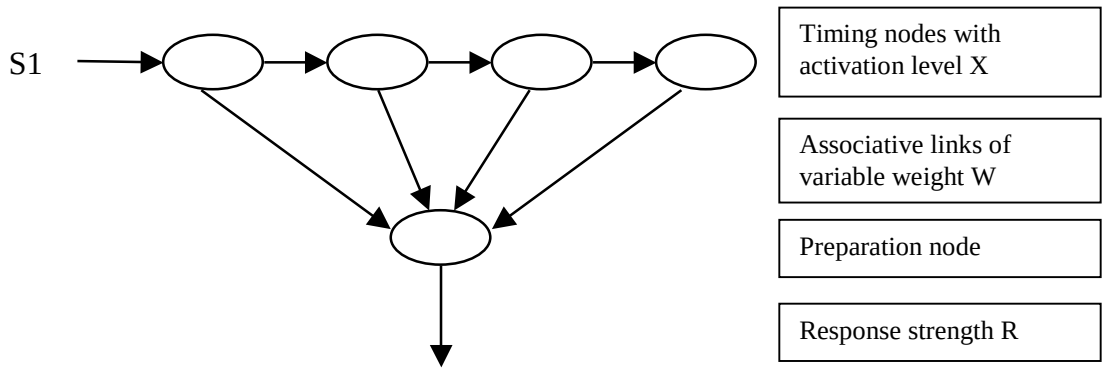


Figure 1. Structure of Machado's conditioning model (adjusted from Machado, 1997)

The weights reflect the state of conditioning, and are adjusted by learning rules, of which the main principles are as follows. First, *during* the foreperiod extinction takes place, which involves the decrease of weights in real time in proportion to the activation of their corresponding timing nodes. Second, *after* the presentation of the imperative stimulus a process of reinforcement takes over, which involves an increase of the weights in accordance with the current activation of their timing nodes, to preserve the importance of the imperative moment. In Machado (1997) the more detailed dynamics of the process are given by a mathematical model (based on linear differential equations), representing the (local) temporal relationships between the variables involved. For example,

$$d/dt\ X(t,n) = \lambda X(t,n-1) - \lambda X(t,n)$$

expresses how the activation level of the n-th timing node $X(t+dt,n)$ at time point t+dt relates to this level $X(t,n)$ at time point t and the activation level $X(t,n-1)$ of the (n-1)-th timing node at time point t. Similarly, as another example,

$$d/dt\ W(t,n) = -\alpha X(t,n)W(t,n)$$

expresses how the n-th weight W(t+dt,n) at time point t+dt relates to this weight W(t,n) at time point t and the activation level X(t,n) of the n-th timing node at time point t.

3. Modelling Dynamic Properties

As discussed above, mathematical models based on differential equations can be used to model local temporal relationships within conditioning processes. However, conditioning processes can also be characterised by temporal relationships of a less local form. As an example, taken from Los and Van Den Heuvel (2001), a dynamic property can be formulated expressing the monotonicity property that 'the response level increases before the critical moment is reached and decreases after this moment'. This is a more global property, relating response levels at any two points in time before the critical moment (or after the critical moment). Therefore it is useful to explore formalisation techniques, as an alternative to differential equations, to express not only for local properties, but also for non-local properties. A second limitation of differential equations is that they are based on quantitative (calculational) relationships, whereas also non-quantitative aspects may play a role (for example, the monotonicity property mentioned above). This suggests that it may be useful to explore alternative formalisation techniques for dynamic properties of conditioning processes that allow one to express both quantitative and non-quantitative aspects.

As already mentioned in the Introduction, the approach presented in this paper indeed introduces alternative formalisation languages to express dynamic properties of conditioning processes, both for local and nonlocal properties and both for quantitative and non-quantitative aspects. These formalisation languages are briefly introduced in the next sections. After this introduction, in a subsequent section first a model based on local properties of a conditioning process is presented (comparable to and inspired by Machado's model).

3.1. Languages to Model Dynamic Properties

The domain of reasoning about dynamical systems in disciplines such as the Behavioural Sciences requires an abstract modelling form yet showing the essential dynamic properties. A high-level language is needed to characterise and formalise dynamic properties of such a dynamical system. To this end the *Temporal Trace Language* TTL is used as a tool; for previous applications of this language to the analysis of (cognitive) processes, (see Jonker and Treur, 2002; Jonker and Treur, 2003a,b; Jonker, Treur, and Vries, 2002). Using this language, dynamic properties can be expresed in informal, semi-formal, or formal format. Moreover to perform simulations, models are desired that can be formalised and are computationally easy to handle. These executable models are based on the socalled '*leads to*' format which is defined as a sublanguage of TTL; for a previous application of this format for simulation of cognitive processes, see (Jonker, Treur, and Wijngaards, 2003). The Temporal Trace Language TTL is briefly defined as follows.

A *state ontology* is a specification (in order-sorted logic) of a vocabulary to describe a state of a process. A state for ontology Ont is an assignment of truth-values true or false to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered, for example the natural or real numbers. A *trace* $\mathcal{T}$ over a state ontology Ont and time frame T is a mapping $\mathcal{T}$: T $\rightarrow$ STATES(Ont), i.e., a sequence of states $\mathcal{T}_t$ ($t \in T$) in STATES(Ont). The set of all traces over

state ontology Ont is denoted by TRACES(Ont). The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

These states can be related to state properties via the formally defined satisfaction relation $\models$, comparable to the Holds-predicate in the Situation Calculus (cf. Reiter, 2001): state($\mathcal{T}$, t) $\models$ p denotes that state property p holds in trace $\mathcal{T}$ at time t. Based on these statements, dynamic properties can be formulated , using quantifiers over time and the usual first-order logical connectives $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\Rightarrow$ (implies), $\forall$ (for all), $\exists$ (there exists); to be more formal: formulae in a sorted first-order predicate logic with sorts T for time points, Traces for traces and F for state formulae.

To model basic mechanisms of a process at a lower aggregation level, direct temporal dependencies between two state properties, the simpler 'leads to' format is used. This executable format can be used for simulation and is defined as follows. Let α and β be state properties. In leads to specifications the notation $\alpha \rightarrow\!\!\!\rightarrow_{e, f, g, h} \beta$ means:

if state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval h.

For a more formal definition, see (Jonker, Treur, and Wijngaards, 2003).

3.2. Types of dynamic properties

Dynamic properties of a conditioning process can be specfied at different levels of aggregation. At the highest level, *global dynamic properties*, i.e., properties of the conditioning process as a whole, can be expressed, for example indicating how a certain pattern of behaviour has been changed by a conditioning process. At the lowest level of aggregation, *local properties* are dynamic properties of the basic mechanisms of the conditioning process. Based on these local properties, and certain conditions of the environment, the global properties of the system emerge. Such conditions of the environment (of the subject) are characterised by *environmental properties*. In laboratory circumstances, these properties are usually guaranteed by a specific experimental design.

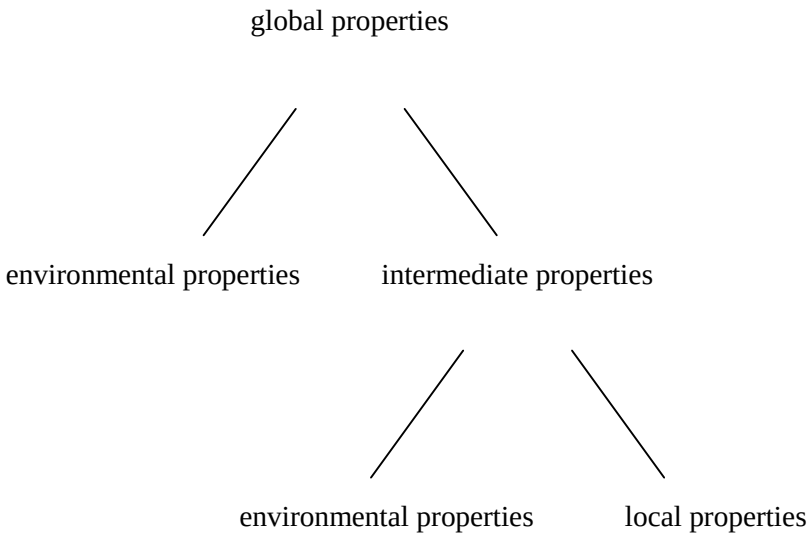


Figure 2. Interlevel relations between dynamic properties at different levels of aggregation

Local properties are logically related to global properties in the sense that the local properties together with the relevant environmental properties entail the global properties. To clarify such logical *interlevel relations*, it is often useful to specify dynamic properties at intermediate levels of aggregation: *intermediate properties*. Thus the overall picture shown in Figure 2 is obtained.

4. Local properties

The local properties (LPs) we defined in order to describe the conditioning process are presented below. These properties are executable dynamic properties (in 'leads to' format) of the elements of this model. Within the dynamic properties the following state properties are used:

X(n,u)	Timing node n has activation level u. In the current simulation, n ranges over the discrete domain [0,5]. Thus, our model consists of six timing nodes. The activation level u can take any continuous value in the domain [0,1].
W(n,v)	Associative link n has weight v. Again, n ranges over the discrete domain [0,5]. The weight v can take any continuous value in the domain [0,1].
R(r)	The preparation node has response strength r (a continuous value in the domain [0,1]).
S1(s)	Warning stimulus S1 occurs with strength s. Within our example, s only takes the values 0.0 and 1.0. However, the model could be extended by allowing any continuous value in-between.
S2(s)	Imperative stimulus S2 occurs with strength s.
Xcopy(n,u)	Timing node n had activation level u at the moment of the occurrence of the last imperative stimulus (S2). See dynamic property LP4 and LP6.
instage(ext)	The process is in a stage of extinction. This stage lasts from the occurrence of S1 until the occurrence of S2.
instage(reinf)	The process is in a stage of reinforcement. This stage starts with the occurrence of S2, and lasts during a predefined reinforcement period, (e.g. 3 seconds).
instage(pers)	The process is in a stage of persistence. This stage starts right after the reinforcement stage, and lasts until the next occurrence of S1.

As Machado (1997)'s model was used as a source of inspiration, for some of the properties presented below the comparable differential equation within Machado's model is given as well. However, since Machado's mathematical approach differs at several points from the logical approach presented in this paper, there is not always a straightforward 1:1 mapping between both formalisations. For instance, state property X(n,u) within our 'leads to' formalisation has a slightly different meaning than the corresponding term X(t,n) in Machado's differential equations. In the former, n stands for the timing node, u stands for the activation level, and X(n,u) stands for the fact that timing node n has activation level u. In the latter, t stands for a time point, n stands for the timing node, and X(t,n) as a whole stands for the activation level.

LP1 Initialisation

The first local property LP1 expresses the initialisation of the values for the timing nodes and the associative links. Formalisation (for n ranging over [0,5]):

start $\twoheadrightarrow$ X(n, 0) $\wedge$ W(n, 0)

LP2 Activation of initial timing nodes
Local property LP2 expresses the activation (and adaptation) of the 0^{th} timing node. Immediately after the occurrence of the warning stimulus (S1), this state has full strength. After that, its value decreases until the next warning stimulus. Together with LP3, this property causes the spread of activation across the timing nodes. Here, $\lambda > 0$ is a rate parameter that controls the speed of this spread of activation, and step is a constant indicating the smallest time step in the simulation. For the simulation experiments presented in the next section, λ was set to 10 and step was set to 0.05. Formalisation:
$X(0, u) \wedge S1(s) \twoheadrightarrow X(0, u*(1-\lambda*step)+s)$
Comparable differential equation in Machado (1997)'s model: $d/dt\ X(t,0) = -\lambda X(t,0)$.

LP3 Adaptation of timing nodes
LP3 expresses the adaptation of the n^{th} timing node (for n ranging over [1,5]), based on its own previous state and the previous state of the $n\text{-}1^{th}$ timing node. Together with LP2, this property causes the spread of activation across the timing nodes. Here, λ is a rate parameter that controls the speed of this spread of activation (see LP2). Formalisation (for n ranging over [1,5]):
$X(n, u1) \wedge X(n-1, u0) \twoheadrightarrow X(n, u1+\lambda*(u0-u1)*step)$
Comparable differential equation in Machado (1997)'s model: $d/dt\ X(t,n) = \lambda X(t,n-1) - \lambda X(t,n)$.

LP4 Storage of timing nodes at moment of reinforcer
LP4 is needed to store the value of the n^{th} timing node at the moment of the occurrence of the imperative stimulus (S2). These values are used later on by property LP6. Formalisation (for n ranging over [0,5]):
$X(n, u) \wedge S2(1.0) \twoheadrightarrow Xcopy(n, u)$

LP5 Extinction of associative links
LP5 expresses the adaptation of the associative links during extinction, based on their own previous state and the previous state of the corresponding timing node. Here, α is a learning rate parameter. For the simulation experiments presented in the next section, the value 2 was chosen for α. Formalisation (for n ranging over [0,5]):
$instage(ext) \wedge X(n, u) \wedge W(n, v) \twoheadrightarrow W(n, v*(1-\alpha*u*step))$
Comparable differential equation in Machado (1997)'s model: $d/dt\ W(t,n) = -\alpha X(t,n)W(t,n)$.

LP6 Reinforcement of associative links
LP6 expresses the adaptation of the associative links during reinforcement, based on their own previous state and the previous state of Xcopy. Here, β is a learning rate parameter. For the simulation experiments presented in the next section, the value 2 was chosen for β. Formalisation (for n ranging over [0,5]):
$instage(reinf) \wedge Xcopy(n, u) \wedge W(n, v) \twoheadrightarrow W(n, v*(1-\beta*u*step) + \beta*u*step)$
Comparable differential equation in Machado (1997)'s model: $d/dt\ W(t,n) = \beta X(T,n)[K-W(t,n)]$.

LP7 Persistence of associative links
LP7 expresses the persistence of the associative links at the moments that there is neither extinction nor reinforcement. Formalisation (for n ranging over [0,5]):
$instage(pers) \wedge W(n, v) \twoheadrightarrow W(n, v)$

LP8 Response function
LP8 calculates the response by adding the discriminative function of all states, i.e., their associative links multiplied by the degree of activation of the corresponding state. Formalisation:
$W(1, v1) \wedge W(2, v2) \wedge W(3, v3) \wedge W(4, v4) \wedge W(5, v5) \wedge X(1, u1) \wedge X(2, u2) \wedge X(3, u3) \wedge X(4, u4) \wedge X(5, u5) \twoheadrightarrow R(v1*u1 + v2*u2 + v3*u3 + v4*u4 + v5*u5)$

LP9 Initialisation of stage pers
LP9 expresses that the initial stage of the process is pers. Formalisation:
$start \twoheadrightarrow instage(pers)$

LP10 Transition to stage ext
LP10 expresses that the process switches to stage ext when a warning stimulus occurs. Formalisation:
$S1(1.0) \twoheadrightarrow instage(ext)$

LP11 Persistence of stage ext
LP11 expresses that the process persists in stage ext as long as no imperative stimulus occurs. Formalisation:
$instage(ext) \wedge S2(0.0) \twoheadrightarrow instage(ext)$

LP12 Transition to stage reinf and pers

LP12 expresses that the process first switches to stage reinf for a while, and then to stage pers when an imperative stimulus occurs. Notice that LP12a and LP12b must have different timing parameters to make sure both stages do not occur simultaneously. Formalisation:

$S2(1.0) \twoheadrightarrow instage(reinf)$ (LP12a)
$S2(1.0) \twoheadrightarrow instage(pers)$ (LP12b)

LP13 Persistence of stage pers

LP13 expresses that the process persists in stage pers as long as no warning stimulus occurs. Formalisation:

$instage(pers) \wedge S1(0.0) \twoheadrightarrow instage(pers)$

5. Simulation Examples

Simulation of executable models is performed by a special software environment. This software environment generates simulation traces of the conditioning process based on an input consisting of dynamic properties in 'leads to' format. An example of such a trace can be seen in Figure 3. Here, time is on the horizontal axis, the relevant state properties (S1, S2, instage(ext), instage(pers), instage(reinf) and R) are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period.

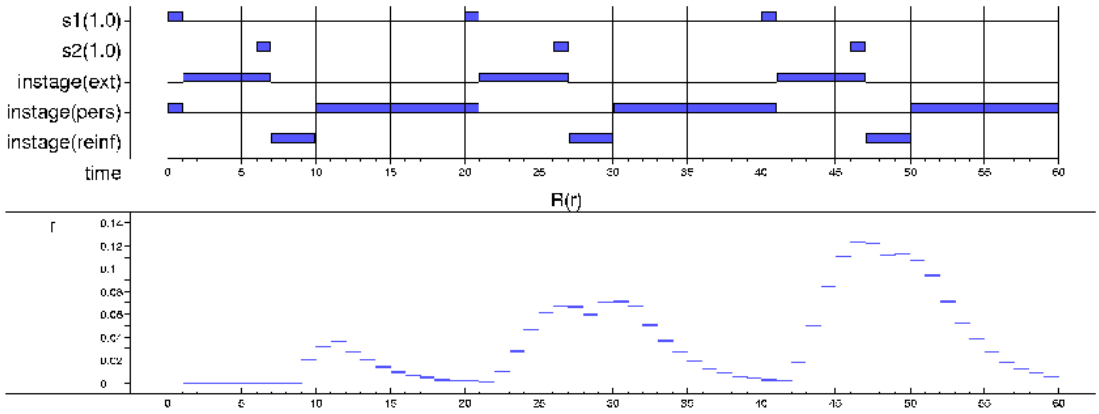


Figure 3. Simulation trace of the dynamics during conditioning (pure block, FP=6)

This trace is based on all local properties presented above. For almost all properties, the timing parameters (0,0,1,1) were used. Exceptions are the properties LP4, LP12a and LP12b. For these properties, the timing parameters were respectively (0,0,1,3), (0,0,1,3) and (3,3,1,1), where 3 corresponds to the reinforcement duration. Notice that we are dealing with a pure block, where the foreperiod is 6 on each trial.

This trace describes the dynamics *during* (not after) the conditioning process. As can be seen in Figure 3, the level of response-related activation increases on each trial. Initially, the subject is not prepared at all: at the moment of the imperative stimulus (S2), the level of response is 0. However, already after two trials a peak in response level has developed that coincides exactly with the imperative moment.

Figure 4 describes the dynamics of the same pure block (with foreperiod 6) *after* the conditioning has taken place. At this moment, the internal model has evolved in such a way that the subject is maximally prepared (response strength $r > 0.4$) at the critical moment, even without the actual occurrence of an imperative stimulus S2.

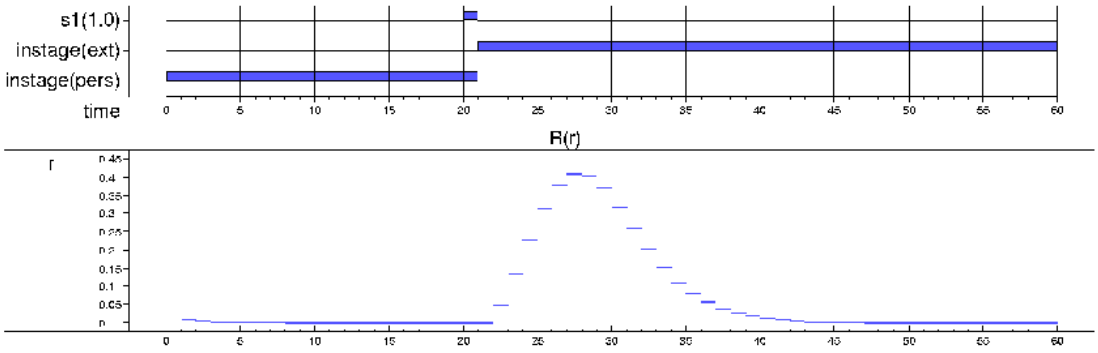


Figure 4. Simulation trace of the dynamics after conditioning
(pure block, FP=6)

In contrast to Figure 3 and 4 (describing the dynamics of a pure block), Figure 5 is an example of a trace where a mixed block is considered. As in Figure 4, we are dealing with a situation where the conditioning has already occurred, but this time, two types of foreperiod (FP=2 and FP=10) have randomly been presented during the preceding trials. As a consequence, the curves that plot the response level have two peaks: one for each critical moment. The current trace shows two trials: one in which the imperative moment corresponds to the first critical moment, and one in which it corresponds to the second critical moment. A detailed explanation of the shape of both curves will be given in the next sections.

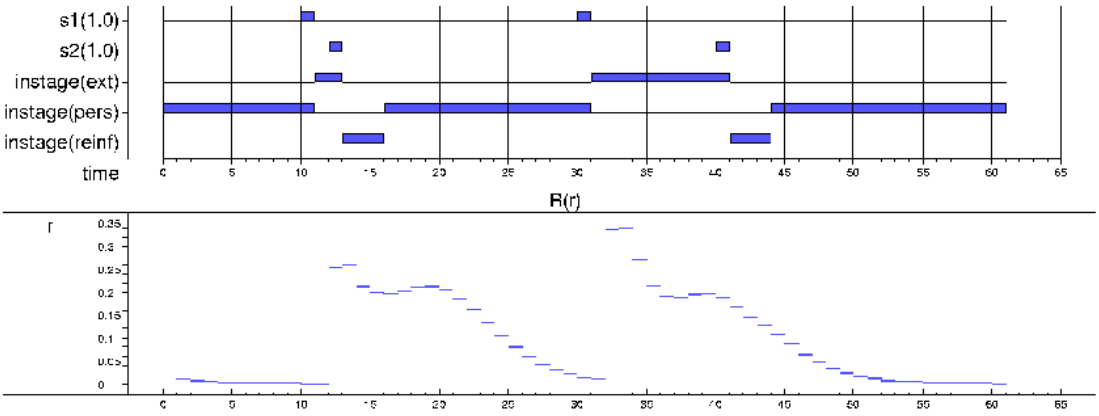


Figure 5. Simulation trace of the dynamics after conditioning
(mixed block, FP=2 and FP=10)

Besides the examples presented here, a large number (about 20) of similar experiments have been performed, with different parameters for foreperiod and block type. The results were consistent with the data produced by Machado.

6. Analysis of Nonlocal Dynamic Properties

In (Los and Van Den Heuvel, 2001), the following dynamic properties of the overall conditioning process are put forward:

> *'Corresponding to each critical moment there is a state of conditioning, the adjustment of which is governed by learning rules of trace conditioning (specified subsequently).'*

(1) 'The state of conditioning implicates an increase and decay of response-related activation as a critical moment is bypassed in time.'

(2) 'the conditioned response takes more time to build up and decay and its corresponding asymptotic value is lower when its corresponding critical moment is more remote from the warning signal.'

(3) 'on any trial, the strength of the conditioned response corresponding to a critical moment is reinforced (i.e., increased toward its asymptote) if and only if that critical moment coincides with the imperative moment.'

(4) 'on any trial the strength of the conditioned response is extinguished (i.e., driven away from its asymptote) if and only if its corresponding critical moment occurs before the imperative moment, whereas it is left unaffected if its corresponding critical moment occurs later than the imperative moment.'

(Los and Van Den Heuvel, 2001, p. 372.)

These properties have a rather informal and non-mathematical nature. Below it is first shown how these properties can be formalised in TTL. In contrast to the earlier presented local properties, we will call these properties global properties (GPs). In the next sections it is shown how these global properties can be checked against the generated traces, and how they can be related to the executable local properties.

GP1 has_global_hill_prep(γ, t1, t2, s1, a, u)
The first global property GP1 is a formalisation of informal property (1) presented above. It describes the following: If at t1 a stimulus s1 starts, then the preparation level for action a will increase from t1 until t2 and decrease from t2 until t1 + u, under the assumption that no stimulus occurs too soon (within u time) after t1. Formally:

$\forall t', t'', s', p', p'', x, x'$
$stimulus\_starts\_at(\gamma, t1, s1, x)$ &
$\neg\ stimulus\_starts\_within(\gamma, t1, t1+u, s', x')$ &
$has\_preparation\_level\_at(\gamma, t', p', a)$ &
$has\_preparation\_level\_at(\gamma, t'', p'', a)$
$\Rightarrow\ [t1 \le t' < t'' \le t2\ \&\ t'' \le t1 + u\ \Rightarrow\ p' < p'']\ \&$
$[t2 \le t' < t'' \le t1 + u\ \Rightarrow\ p' > p'']$

GP2 pending_peak_versus_critical_moment(γ1, γ2, t1, t2, c1, c2)
Global property GP2 is a formalisation of informal property (2). It describes that: If for trace γ2 at time t2 peak time c2 is more remote than peak time c1 for γ1 at time t1, then at t2 in γ2 the pending peak level is lower than the pending peak level at t1 in γ1. Formally:

$\forall\ s1, a, p1, p2$
$has\_pending\_peak\_level(\gamma1, t1, c1, p1, s1, a)$ &
$has\_pending\_peak\_level(\gamma2, t2, c2, p2, s1, a)$
$\Rightarrow\ [\ c1 < c2\ \Rightarrow\ p1 > p2\ ]$

GP3 dynamics_of_pending_preparation(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε)
GP3 is a formalisation of both informal property (3) and (4) together. It describes that:
If t1 < t2
and at t1 the pending preparation level for time t1+v, action a, and stimuli s1 and s2 is p,
and at t2+d the pending preparation level for time t2+d+v, action a, and stimuli s1 and s2 is p',
and in trace γ at time t1 a stimulus s1 starts,
and in trace γ at time t2 a stimulus s2 starts,
and in trace γ the maximum peak level for a is pmax,
and in trace γ the minimum preparation level for a is pmin,
then:

$t2 \in [\ t1 + c - \varepsilon,\ t1 + c + \varepsilon\ ]$	iff	$p' > p$	(reinforcement, given that $p < pmax$)
$t2 > t1 + c + \varepsilon$	iff	$p' < p$	(extinction, given that $p > pmin$)
$t2 < t1 + c - \varepsilon$	iff	$p' = p$	(persistence)

Parameter d refers to the time needed to process the events (d > 0), and c refers to a critical moment. Formally:

dynamics_of_pending_preparation(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε) ⇔
  reinforcement(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε) &
  extinction(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε) &
  persistence(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε)

reinforcement(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε) ⇔
  ∀ x1, x2, pmin, pmax
  two_stimuli_occur(γ, t1, t2, c, v, p, p', s1, s2, a, d)
  ⇒ [p < pmax ⇒ [t2 ∈ [t1 + c - ε, t1 + c + ε] ⇔ p' > p]]

extinction(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε) ⇔
  ∀ x1, x2, pmin, pmax
  two_stimuli_occur(γ, t1, t2, c, v, p, p', s1, s2, a, d)
  ⇒ [p > pmin ⇒ [t2 > t1 + c + ε ⇔ p' < p]]

persistence(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε) ⇔
  ∀ x1, x2, pmin, pmax
  two_stimuli_occur(γ, t1, t2, c, v, p, p', s1, s2, a, d)
  ⇒ [t2 < t1 + c - ε ⇔ p' = p]

two_stimuli_occur(γ, t1, t2, c, v, p, p', s1, s2, a, d) ⇔
  t1 < t2 & has_pending_preparation_level(γ, t1, t1+v, p, s1, s2, a) &
  has_pending_preparation_level(γ, t2+d, t2+d+v, p', s1, s2, a) &
  stimulus_starts_at(γ, t1, s1, x1) &
  stimulus_starts_at(γ, t2, s2, x2) &
  target_action_for(a, s2) &
  is_a_critical_moment(c) &
  maximum_peak_level(γ, pmax, a) &
  minimum_preparation_level(γ, pmin, a)

7. Checking Nonlocal Properties on Traces

In addition to the software described in the Simulation section, other software has been developed that takes traces and formally specified properties as input and checks whether a property holds for a trace. Using automatic checks of this kind, the four formalised properties based on (Los and Van Den Heuvel, 2001) have been checked against the traces depicted in Figure 3 to 5. This section discusses the results of these checks.

GP1 has_global_hill_prep(γ, t1, t2, s1, a, u)
This property has been checked against several traces. An interesting observation concerning these checks was the fact that the property does not hold for traces that describe the dynamics *during* conditioning (like the trace in Figure 3). From this, we may conclude that in these traces it is not the case that each critical moment corresponds to a single peak in response level.

However, when checking against traces describing the dynamics *after* conditioning (e.g. Figure 4 and 5), the property does hold, as long as the parameters are well chosen. In particular, the parameters should meet the following conditions:

- γ = a trace describing the dynamics after a conditioning process (pure block or mixed block)
- t1 = a time point when s1 occurs
- t2 = t1+FP (where FP is the foreperiod during the preceding conditioning process)

- s1 = the warning stimulus
- a = the action for which the subject prepares after observing s1
- u = iti (the intertrial interval during the preceding conditioning process)

To give an example, the following property meets these conditions: has_global_hill_prep(γ1, 20, 27, s1, a, 20), where γ1 is the trace provided in Figure 4. Thus, for this trace the following holds: if at time point 20 a stimulus s1 starts, then the preparation level for action a increases from 20 until 27 and decreases from 27 until 40, under the assumption that no stimulus occurs between 20 and 40.

GP2 pending_peak_versus_critical_moment(γ1, γ2, t1, t2, c1, c2)

Checking property GP2 involves comparing two traces. Basically, it states that in traces where the foreperiod is longer, the level of response is lower. In order to check GP2, several traces have been generated that are similar to the trace in Figure 4, but each with a different foreperiod. For all combinations of traces, the property turned out to hold, but again, the choice of the parameters was limited by certain constraints:

- γ1 = a trace describing the dynamics after a conditioning process (pure block or mixed block)
- γ2 = a trace describing the dynamics after a conditioning process (pure block or mixed block)
- t1 = a time point
- t2 = t1
- c1 = the peak time for trace γ1 at time t1
- c2 = the peak time for trace γ2 at time t2

As an extra condition, note that the number of trials and the block type of γ1 should be similar to those of γ2. Otherwise, it makes no sense to compare both traces. E.g., the following property holds: pending_peak_versus_critical_moment(γ1, γ2, 20, 20, 6, 7), where γ1 is the trace provided in Figure 4, and γ2 is a similar trace with FP=7. This means that, if for trace γ2 at time 20 peak time 7 is more remote than peak time 6 for γ1 at time 20 (which is indeed the case), then at 20 in γ2 the pending peak level is lower than the pending peak level at 20 in γ1 (which is also the case).

GP3 dynamics_of_pending_preparation(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε)

Property GP3 combines property (3) and (4) as mentioned in the previous section. Basically, the property consists of three separate statements that relate the strength of the conditioned response (p) to the critical moment (t1+c) and the imperative moment (t2), by stating that:

a) p increases iff t2 = t1+c

b) p decreases iff t2 > t1+c

c) p remains the same iff t2 < t1+c

Also for this property, it is important to choose the parameters with care. They should meet the following conditions in order for the property to make sense:

- γ = a trace describing the dynamics after a conditioning process (pure block or mixed block)
- t1 = a time point when S1 occurs

- t2 = a time point when S2 occurs
- c = a foreperiod within γ (such that c = t2-t1)
- v = c
- s1 = the warning stimulus
- s2 = the imperative stimulus
- a = the action for which the subject prepares after observing s1
- d = long enough to process the events (e.g. iti)
- ε = 0

An example of a property that meets these criteria is: dynamics_of_pending_preparation(γ, 10, 12, 10, 10, p, p', s1, s2, a, 18, 0), where γ is the trace depicted in Figure 5. However, even with these parameter settings the property turned out not to hold. A close examination of Figure 5 will reveal the cause of this failure. This trace describes a mixed block with two types of foreperiod (FP=2 and FP=10). At time point 10, a warning stimulus (S1) occurs. At this time point, the pending preparation level for the latest critical moment (time point 20) has a certain value. And since this critical moment occurs after the occurrence of S2 (the imperative moment: time point 12), the pending preparation level for the latest critical moment should remain the same, according to statement C above. However, in the trace in question this is not the case (see Figure 5: in the second curve the second peak is slightly lower than in the first curve). Hence, it may be concluded that statement C (sub-property persistence presented earlier) does not hold for the chosen parameters.

Summarising, automated checks have pointed out that property GP3 does not always hold for traces generated by our simulation model. In particular, in traces where the critical moment occurs after the imperative moment, the strength of the conditioned response does not stay exactly the same. Additional tests have indicated that this value sometimes decreases, and sometimes increases a bit, depending on the specific settings. Fortunately, an explanation of this finding can be found in a later section of (Los and Van Den Heuvel, 2001), where the authors revise their original model as follows:

> *'According to the original model, extinction and reinforcement affect each state of conditioning in an all-or-none way, thereby excluding a coupling between states of conditioning corresponding to adjacent critical moments. According to the revised model, extinction and reinforcement affect the states of conditioning more gradually across the time scale, resulting in a coupling between adjacent states.' (Los and Van Den Heuvel, 2001), p. 383.*

The revision of the model also implies a revision of property GP3. To be more specific, sub-property persistence can be changed into the following:

persistence(γ, t1, t2, c, v, p, p', s1, s2, a, d, ε) ⇔
  ∀ x1, x2, pmin, pmax
  two_stimuli_occur(γ, t1, t2, c, v, p, p', s1, s2, a, d)
  ⇒ [t2 < t1 + c - ε ⇔ p' ∈ [p - δ, p + δ]]

Here, δ is a tolerance factor allowing a small deviation from the strength of the original response. After adapting GP3 accordingly, the property turned out to hold.

8. Discussion

Two software environments have been developed to support the research reported here. First a simulation environment has been used to generate simulation traces as shown. Second, checking software has been used that takes traces and formally specified properties and checks whether a property holds for a trace.

In comparison to Executable Temporal Logic (Barringer et al., 1996) our simulation approach has possibilities to incorporate (real or integer) numbers in state properties, and in the timing parameters e, f, g, h. Similarly, our approach to analysis has higher expressiveness than approaches in temporal logic such as (Fisher and Wooldridge, 1997).

The present paper has confirmed, by means of formal verification, that the assumptions of the informal conditioning model proposed by Los and Van den Heuvel (2001) are global properties of the formal model developed by Machado (1997), given certain restrictions of the parameter values, and given slight adaptations of the persistence rule given by GP3C. This is an important finding, because the global properties have proved to be highly useful in accounting for key findings in human timing (Los & Van den Heuvel, 2001; Los et al., 2001).

One crucial finding the global properties can deal with effectively is the occurrence of sequential effects of FP. These effects entail that on any given trial, RT is longer when the FP of that trial is shorter than the FP of the preceding trial relative to when it is as long as or longer than the FP of the preceding trial. Stated differently, RT is longer when the imperative moment was bypassed during the FP on the preceding trial than when it was not bypassed during FP on the preceding trial (e.g., Los & Van den Heuvel, 2001; Niemi & Naatanen, 1981). This finding is well accounted for by the learning rules formulated as GP3. According to GP3B, the state of conditioning (p) associated with a critical moment is subject to extinction when a critical moment is bypassed during FP (i.e., $t2 > t1 + c$), which is neither the case for the imperative moment, where according to GP3A reinforcement occurs (i.e., $t2 > t1 + c$), nor for critical moment beyond the imperative moment, where the state of conditioning persists according to GP3C (i.e., $t2 < t1 + c$). Note that the adjustment of GP3C suggested by the present check of nonlocal properties does not compromise the effectiveness of these learning rules, because the tolerance factor δ is small relative to the extinction described by GP3B.

In fact, the addition of the tolerance area δ to GP3C, might prove to be helpful in accounting for a more subtle effect in the extant literature. This concerns the finding that the FP-RT functions obtained in pure and mixed blocks cross over at the latest critical moment. Specifically, in pure blocks, the FP – RT function has been found to be upward sloping, given a minimal FP of about 250 – 300 ms. By contrast, in mixed blocks, the RT is slowest at the shortest critical moment (due to the influence of sequential effects described in the previous paragraph) and decreases as a negatively accelerating function of FP. At the latest critical moment the pure and mixed FP – RT functions come together, presumably because this moment is never bypassed during FP on the preceding trial, allowing the state of conditioning to approach its asymptotic value in either case. Sometimes, though, a cross-over of the two FP – RT functions is reported, which has been shown to be particularly pronounced in certain clinical populations, such as people diagnosed with schizophrenia (see Rist & Cohen, 1991, for a review). This finding may be related to the failure to confirm GP3C without the allowance of a tolerance area δ. Thus, it could be that, for certain parameter settings, the state of conditioning corresponding to the latest critical moment approaches its asymptotic value more closely when a shorter FP occurred on the preceding trial (which is often the case in mixed

blocks) than when the same FP occurred on the preceding trial (as is always the case in pure blocks).

In future work, our simulation model will be compared more thoroughly with empirical data. Since the checking software can take traces of different format as input, it will be possible to verify the formalised global properties against experimental human conditioning traces.

Acknowledgements

Lourens van der Meij provided the technical support for the software environments used in this project.

References

Ashby, R. (1960). *Design for a Brain*. Second Edition. Chapman & Hall, London.

Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M. (1996). *The Imperative Future: Principles of Executable Temporal Logic*, Research Studies Press Ltd. and John Wiley & Sons.

Busemeyer, J., and Townsend, J.T. (1993). Decision field theory: a dynamic-cognitive approach to decision making in an uncertain environment. *Psychological Review*, vol. 100, pp. 432-459.

Fisher, M., and Wooldridge, M. (1997) On the Formal Specification and Verification of Multi-Agent Systems. International Journal of Cooperative Information Systems, vol. 6, pp. 67-94.

Gallistel, C.R. and Gibbon, J. (2000). Time, rate, and conditioning. *Psychological Review*, vol. 107, pp. 289-344.

Gelder, T.J. van, and Port, R.F., (1995). It's About Time: An Overview of the Dynamical Approach to Cognition. In: (Port and van Gelder, 1995), pp. 1-43.

Jonker, C.M., and Treur, J. (2002). Analysis of the Dynamics of Reasoning Using Multiple Representations. In: W.D. Gray and C.D. Schunn (eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci 2002*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2002, pp. 512-517.

Jonker, C.M., and Treur, J. (2003a). A Temporal-Interactivist Perspective on the Dynamics of Mental States. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 137-155.

Jonker, C.M., and Treur, J. (2003b). Modelling the Dynamics of Reasoning Processes: Reasoning by Assumption. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 119-136.

Jonker, C.M., Treur, J., and Vries, W. de (2002). Temporal Analysis of the Dynamics of Beliefs, Desires, and Intentions. *Cognitive Science Quarterly*, vol. 2, pp. 471-494.

Jonker, C.M., Treur, J., and Wijngaards, W.C.A. (2003). A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. *Cognitive Systems Research Journal*, vol. 4(3), 2003, pp. 191-210.

Kowalski, R., and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4:67-95, 1986.

Los, S.A., Knol, D.L., and Boers, R.M. (2001). The Foreperiod Effect Revisited: Conditioning as a Basis for Nonspecific Preparation. *Acta Psychologica*, vol. 106, pp. 121-145.

Los, S.A., and Van Den Heuvel, C.E. (2001). Intentional and Unintentional Contributions to Nonspecific Preparation During Reaction Time Foreperiods. *Journal of Experimental Psychology: Human Perception and Performance*, vol. 27, pp. 370-386.

Machado, A. (1997). Learning the temporal Dynamics of Behaviour. *Psychological Review*, vol. 104, pp. 241-265.

Niemi, P and Naatanen, R. (1981). Foreperiod and Simple Reaction Time. *Psychological Bulletin*, vol 89, pp. 133-162.

Port, R.F., Gelder, T. van (eds.) (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, Mass.

Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

Rist, F. and Cohen, C. (1991). Sequential effects in the reaction times of schizophrenics: crossover and modality shift effects. In E.R. Steinhauer, J.H. Gruzelier, & J. Zubin, *Handbook of schizophrenia*, vol. 5: Neuropsychology, psychophysiology and information processing (pp. 241-271). Amsterdam: Elsevier.

PART IV ADAPTIVITY

CHAPTER 11

Analysis of Adaptive Dynamical Systems for Eating Regulation Disorders

This chapter appeared as Bosse, T., Delfos, M.F., Jonker, C.M., and Treur, J. (2003). Analysis of Adaptive Dynamical Systems for Eating Regulation Disorders. *Proceedings of the 25th Annual Conference of the Cognitive Science Society, CogSci'03*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Analysis of Adaptive Dynamical Systems for Eating Regulation Disorders

Tibor Bosse[1], Martine F. Delfos[2], Catholijn M. Jonker[1], and Jan Treur[1,3]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{tbosse, jonker, treur}
[2] PICOWO, Psychological Institute for Consultancy, Education and Research,
Goeree 18, 3524 ZZ Utrecht, The Netherlands
[3] Universiteit Utrecht, Department of Philosophy,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Abstract. To analyse a subject's mental processes, psychotherapists often face nontrivial properties of adaptive dynamical systems. Analysis of dynamical systems usually is performed using mathematical techniques. Such an analysis is not precisely the type of reasoning performed in psychotherapy practice. In this paper it is shown how practical reasoning about dynamic properties of adaptive dynamical systems within psychotherapy can be described using dynamical logical methods and a high-level language to describe dynamics.

1. Introduction

Within the context of psychotherapy often types of human behaviour and development are addressed that are highly complex, dynamic and adaptive. Recently it has been suggested that the Dynamical Systems Theory (DST), cf. Port and van Gelder (1995), could be an adequate tool for psychotherapists to describe and analyse such behaviours; e.g., (Kupper and Hoffmann, 1996; Levine, 1996; Tschacher, Scheier, and Grawe, 1998; Warren, Sprott, and Hawkins, 2002). However, application of the DST approach in the practice of psychotherapy is not at all straightforward, and much remains to be done. A therapist's reasoning usually is performed in an informal, intuitive, partly conscious manner. Explanation of (at least parts of) this reasoning may take place in a qualitative, logical manner. In contrast, DST requires quantitative mathematical modelling, and analysis of dynamic properties is based on quantitative techniques from mathematics. This contrast between 'qualitative, logical' and 'quantitative, mathematical' makes it very difficult, if not impossible to use the DST approach in this domain. The main contribution of this paper is to show how alternative techniques are better suited to adequately describe the manner in which reasoning about such an adaptive dynamical system in therapy practice takes place, or can take place in a systematic manner.

Within the areas of Computer Science and Artificial Intelligence recently alternative techniques have been developed to analyse the dynamics of phenomena using logical means. Examples are dynamic and temporal logic, and event and situation calculus; e.g., (Reiter, 2001). These logical techniques allow to consider and relate states of a process at different points in time. The form of these relations can cover qualitative aspects, but also quantitative aspects.

This paper illustrates the usefulness of such an alternative approach for the analysis and formalisation of adaptive dynamical systems in psychotherapy practice, in particular for the first phase of eating regulation disorders; e.g., (Beument et al., 1987; Garner and

Garfinkel, 1985). In Delfos (2002), an adaptive dynamical model that describes normal functioning of eating regulation under varying metabolism levels is used as a basis for classification of eating regulation disorders, and of diagnosis and treatment within a therapy. Reasoning about the dynamic properties of this model (and disturbances of them) is performed in an intuitive, conceptual but informal manner.

In this paper, first this model is formalised in a high-level executable format, and some simulations are shown, both for wellfunctioning situations and for different types of malfunctioning situations that correspond to the first phase of well-known disorders such as anorexia (nervosa), obesitas, and bulimia. Next, as part of our analysis a number of relevant dynamic properties of this dynamical system are identified and formalised at different levels of aggregation: both for the regulation as a whole and for separate parts of the adaptive system. Using a software environment that has been developed, these properties have been checked for a number of simulation traces. Moreover, it is shown how these dynamic properties logically relate to each other, i.e., which properties at the lower level of aggregation together imply given properties at the higher level. Such logical relationships are especially important for the diagnosis of a malfunctioning system.

2. Modelling Approach

The domain of reasoning about dynamical systems in psychotherapy requires an abstract modelling form yet showing the essential dynamic properties. As dynamic properties of such a dynamical system can be complex, a high-level language is needed to characterise them. To this end the *Temporal Trace Language* TTL is used as a tool; for previous applications of this language to the analysis of (cognitive) processes, see, e.g., (Jonker and Treur, 2002). Using this language, dynamic properties can be expressed in informal, semi-formal, or formal format. The language allows to explicitly refer to (real) time, and to developments of processes over time. Moreover to perform simulations, models are desired that can be formalised and are computationally easy to handle. These executable models are based on the socalled '*leads to*' format which is defined as a sublanguage of TTL; for a previous application of this format for simulation of cognitive processes, see (Jonker, Treur, and Wijngaards, 2003). The Temporal Trace Language TTL is briefly defined as follows.

A *state ontology* is a specification (in order-sorted logic) of a vocabulary to describe a state of a process. A state for ontology Ont is an assignment of truth-values true or false to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered, for example the natural or real numbers. A *trace* $\mathcal{T}$ over a state ontology Ont and time frame T is a mapping $\mathcal{T}$: T $\rightarrow$ STATES(Ont), i.e., a sequence of states $\mathcal{T}_t$ ($t \in T$) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

These states can be related to state properties via the formally defined satisfaction relation $\models$, comparable to the Holds-predicate in the Situation Calculus; cf. (Reiter, 2001): state($\mathcal{T}$, t) $\models$ p denotes that state property p holds in trace $\mathcal{T}$ at time t. Based on these statements, dynamic properties can be formulated , using quantifiers over time and the usual first-order logical connectives $\neg$ (not), & (and), $\vee$ (or), $\Rightarrow$ (implies), $\forall$ (for all),

∃ (there exists); to be more formal: formulae in a sorted first-order predicate logic with sorts T for time points, Traces for traces and F for state formulae.

To model basic mechanisms of a process at a lower aggregation level, direct temporal dependencies between two state properties, the simpler 'leads to' format is used. This executable format can be used for simulation and is defined as follows. Let α and β be state properties. In leads to specifications the notation $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$ means:

if state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval h.

For a more formal definition, see (Jonker, Treur, and Wijngaards, 2003).

3. Local properties

Local properties are dynamic properties of the basic mechanisms in the dynamical model. Based on these properties the global properties of the system emerge; they together entail these global properties. Local properties are specified in the executable 'leads to' format; for simplicity, below the parameters e, f, g, and h have been left out (their values are discussed in the section on Simulation Experiments). An overall picture of the executable model can be found in Figure 1.

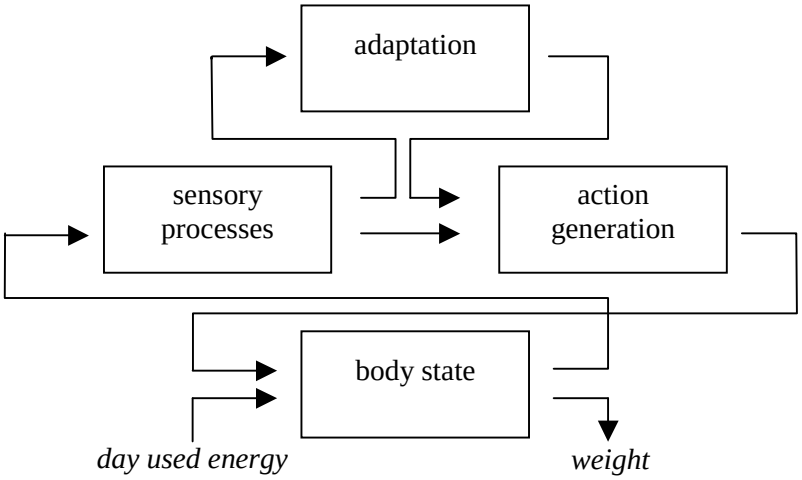


Figure 1. Overview of the executable model

The first two (*action generation*) properties characterise when a stimulus to eat is generated, based on an internal eat norm N that is maintained.

LP1 (eat-stimulus)
The first local property LP1 expresses that an eat norm N and an intermediate amount eaten E less than this norm together lead to an eat stimulus. Formalisation:
intermediate_amount_eaten(E) and eat_norm(N) and E < N $\rightarrow\!\!\!\!\rightarrow$ stimulus(eat)

LP2 (not-eat-stimulus)
Local property LP2 expresses that an eat norm N and an intermediate amount eaten E higher than this norm together lead to an non-eat stimulus. Formalisation:
intermediate_amount_eaten(E) and eat_norm(N) and E ≥ N $\rightarrow\!\!\!\!\rightarrow$ stimulus(do_not_eat)

The properties LP3, LP4, LP5 and LP6 characterise the effect of eating (on *body state*); it is assumed that the outcomes on amount eaten are taken by *sensory processing*.

LP3 (increase of amount eaten)
Local property LP3 expresses how an eat stimulus increases an intermediate amount eaten by additional energy d (the energy value of what is eaten). Formalisation:
intermediate_amount_eaten(E) and stimulus(eat) $\rightarrow\!\!\!\rightarrow$ intermediate_amount_eaten(E+d)

LP4 (stabilizing amount eaten)
Local property LP4 expresses how a non-eat stimulus keeps the intermediate amount eaten the same. Formalisation:
intermediate_amount_eaten(E) and stimulus(do_not_eat) $\rightarrow\!\!\!\rightarrow$ intermediate_amount_eaten(E)

LP5 (day amount eaten)
Local property LP5 expresses that the day amount eaten is the intermediate amount eaten at the end of the day. Formalisation:
intermediate_amount_eaten(E) and time(24) $\rightarrow\!\!\!\rightarrow$ day_amount_eaten(E)

Here time counts the hours from 1 to 24 during the day.

LP6 (weight through balance of amount eaten and energy used)
Local property LP6 expresses a simple mechanism of how weight is affected by the day balance of amount eaten and energy used. Here γ is a fraction that specifies how energy leads to weight kilograms. Formalisation:
day_amount_eaten(E1) and day_used_energy(E2) and weight(W) $\rightarrow\!\!\!\rightarrow$ weight(W + γ * (E1 – E2))

The last local property characterises *adaptation*: how the eat norm N is adapted to the day used energy.

LP7 (adaptation of amount to be eaten)
Local property LP7 expresses a simple (logistic) mechanism for the adaptation of the eat norm based on the day amount of energy used. Here α is the adaptation speed, β is the fraction of E that is the limit of the adaptation; normally $\beta = 1$. Formalisation:
day_used_energy(E) and eat_norm(N) and time(24) $\rightarrow\!\!\!\rightarrow$ eat_norm(N + α * N * (1 - N/βE))

4. Simulation Examples

A special software environment has been created to enable the simulation of executable models. Based on an input consisting of dynamic properties in 'leads to' format, the software environment generates simulation traces. Examples of such traces can be seen in Figure 2, 4 and 5. Here, time is on the horizontal axis, the state properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false. These traces are based on all local properties presented above.

Certain parameters are the same in all three simulations. In the properties LP1 to LP5, the values (0,0,1,1) have been chosen for the timing parameters e, f, g, and h. In the properties LP6 and LP7, these values are (0,0,1,25); moreover, $\gamma = 0.2$ in LP6. The initial weight is always 60, the initial eat-norm is always 6, and the amount of energy used on each day remains 8. Thus, we are dealing with situations where initially the eat-norm is too low with respect to the energy used, and should be adapted accordingly. All simulations involve a period of 110 hours (i.e., slightly more than four days). In Figure 2, an example of a normal situation is shown (i.e., no eating regulation disorders are present). To simulate this, in the Norm Adaptation Property (LP7), $\alpha = 0.75$ and $\beta = 1$; As can be seen in the figure, it takes some time before the eat-norm is correctly adapted to the amount of energy used, but in the end they are practically equal. As a consequence, the subject first undereats a little bit (6 units), causing a loss of 0.4 kilogram. However, within the next 24 hours she starts eating more (8 units). Subsequently, the eating pattern stabilizes, and so does the weight (at 59.6 kg).

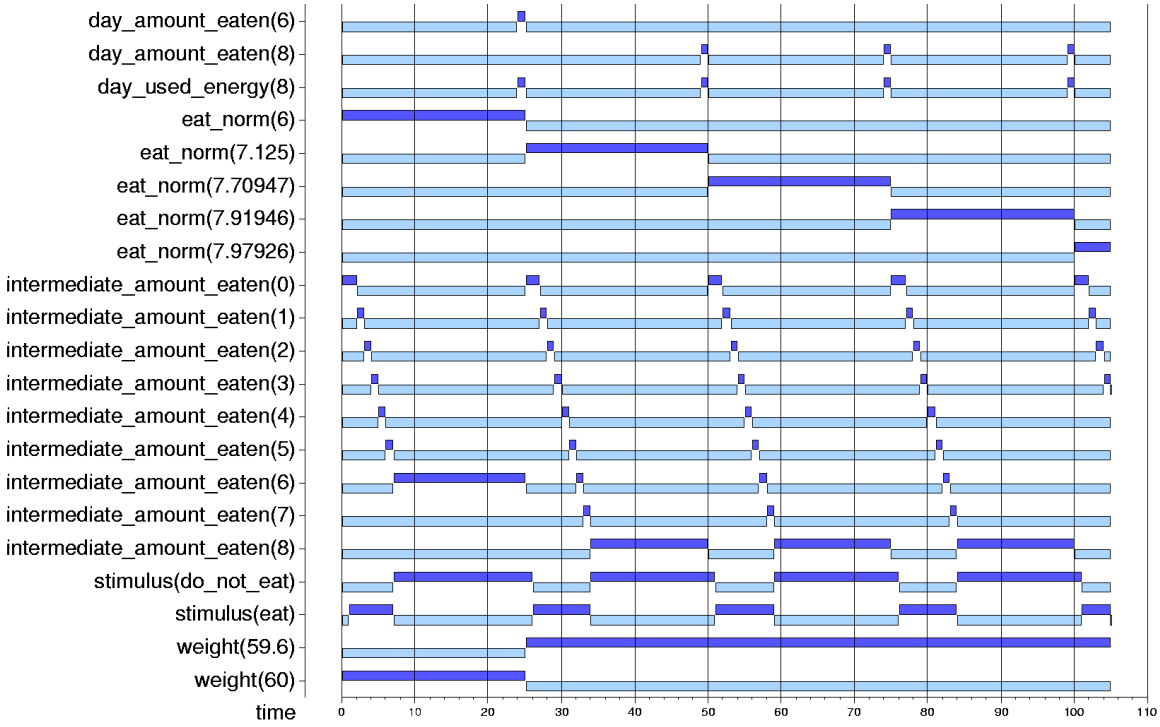


Figure 2. Simulation of a normal pattern

The simulation of anorexia is based upon the assumption that anorexia in many cases has a genetic background (Vink et al., 2001). This means that the signal 'stop eating', in this simulation translated into the 'stimulus(do-not-eat)', comes too early with respect to the amount of energy deployed. Delfos (2002) proposes that as a result of this condition, there exists an unconscious phase of slight underfeeding resulting in not gaining weight proportional to the growth and the risk of hampering growth. This first phase of anorexia, which can cover several years especially prepuberty, consists of a discrepancy between food eaten and energy deployed at an unconscious level; the person is not consciously trying to lose weight.

In Figure 3 the anorexia process is depicted in height velocity (cm/year). The girl entered the conscious phase of her eating disorder (anorexia) when she was nearly 13 years old. It was then that she began dieting. Within a year she was in a very bad medical condition. The height velocity however shows that the growth was stopped much earlier by a delay of puberty from age 10 on. After entering therapy when 14 years old, the height velocity recovered with the process of gaining weight.

Figure 4 shows a simulation of the eating pattern of a person within the first (unconscious) phase of anorexia. To simulate this, in the Norm Adaptation Property, α = 0.75 and β = 0.95. These settings result in an eat norm that converges a little bit to the amount of energy used, but this adaptation is not enough. The picture clearly demonstrates the consequences: the subject continuously eats an amount of food that is too low, compared to what she needs. Therefore, weight drops from 60 to 59.6 to 59.4, and this decreasing trend continues. A simulation of the dynamics of obesitas that has been performed (not shown) provides exactly the opposite pattern. In that case, the simulated subject continuously eats too much and gains weight.

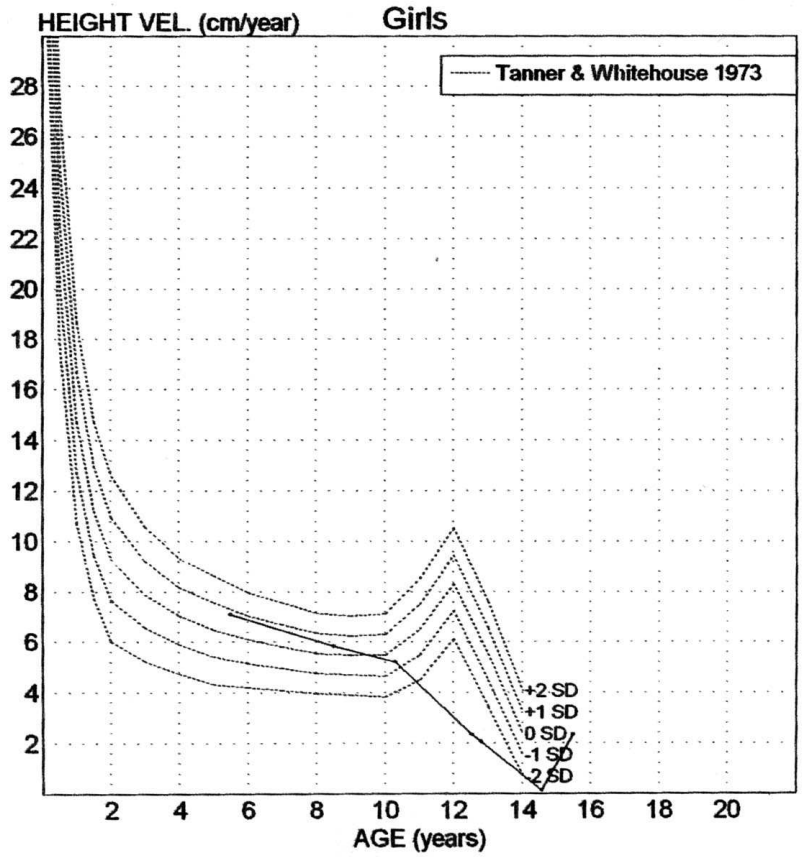


Figure 3. Height velocity pattern for anorexia

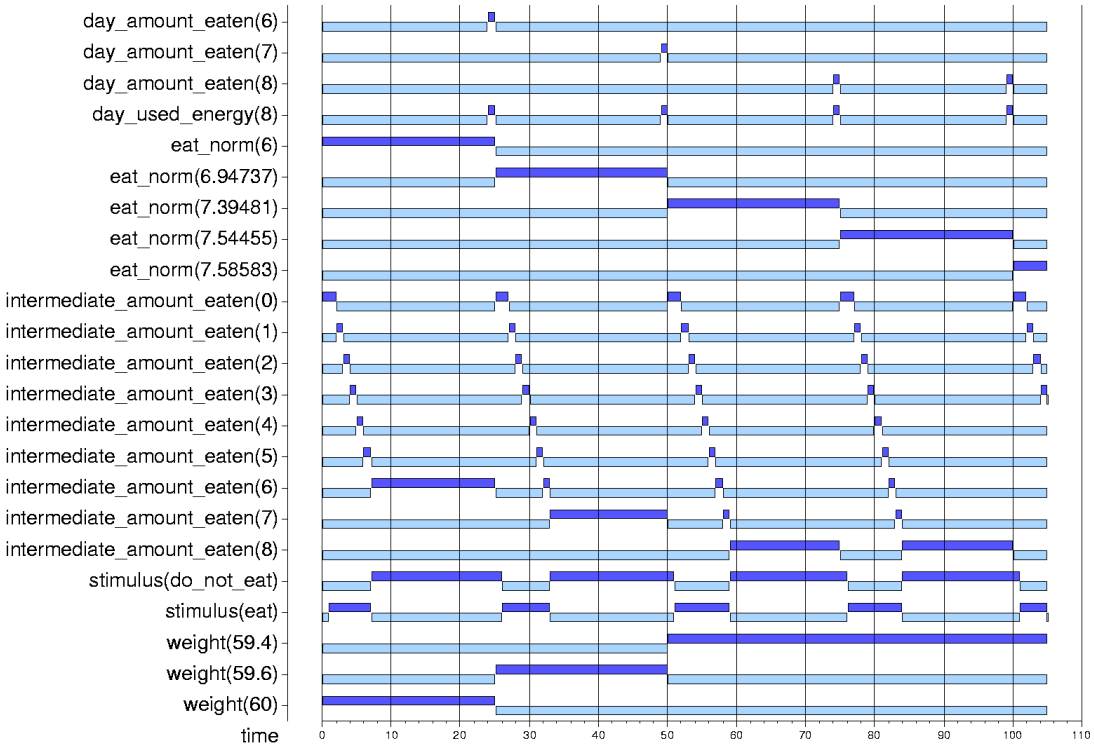


Figure 4. Simulation of the pattern of a person with anorexia

As for bulimia there exists two kinds of situations. First the prephase of bulimia, in which the eating disorder exists at an unconscious level, and second the bulimia that evolves from consciously slight underfeeding or anorectic underfeeding that results in compensating urges of excessive eating.

In Figure 5, a simulation of the eating pattern of a person in a prephase for bulimia is shown. To simulate this, in the Norm Adaptation Property, $\alpha = 2.25$ and $\beta = 1.2$. Especially the value of α is very important here, because it makes that the adaptation of the eat norm to the energy use is too sensitive. Thus, a norm that is too low will be increased, but this increment will be too big, so that the new norm is too high. This behavior can be seen in Figure 5, where the eat-norm keeps fluctuating somewhere between 6 and 12. This results in a very irregular eating pattern. Accordingly, the subject's weight fluctuates between 59 and 62. The risk of developing bulimia fully in the form as known in psychotherapy is present, and will become manifest as soon as the subject starts to attempt to correct these fluctuations by conscious decisions. This further interference of more conscious cognitive aspects within the adaptive processes will be addressed in future research.

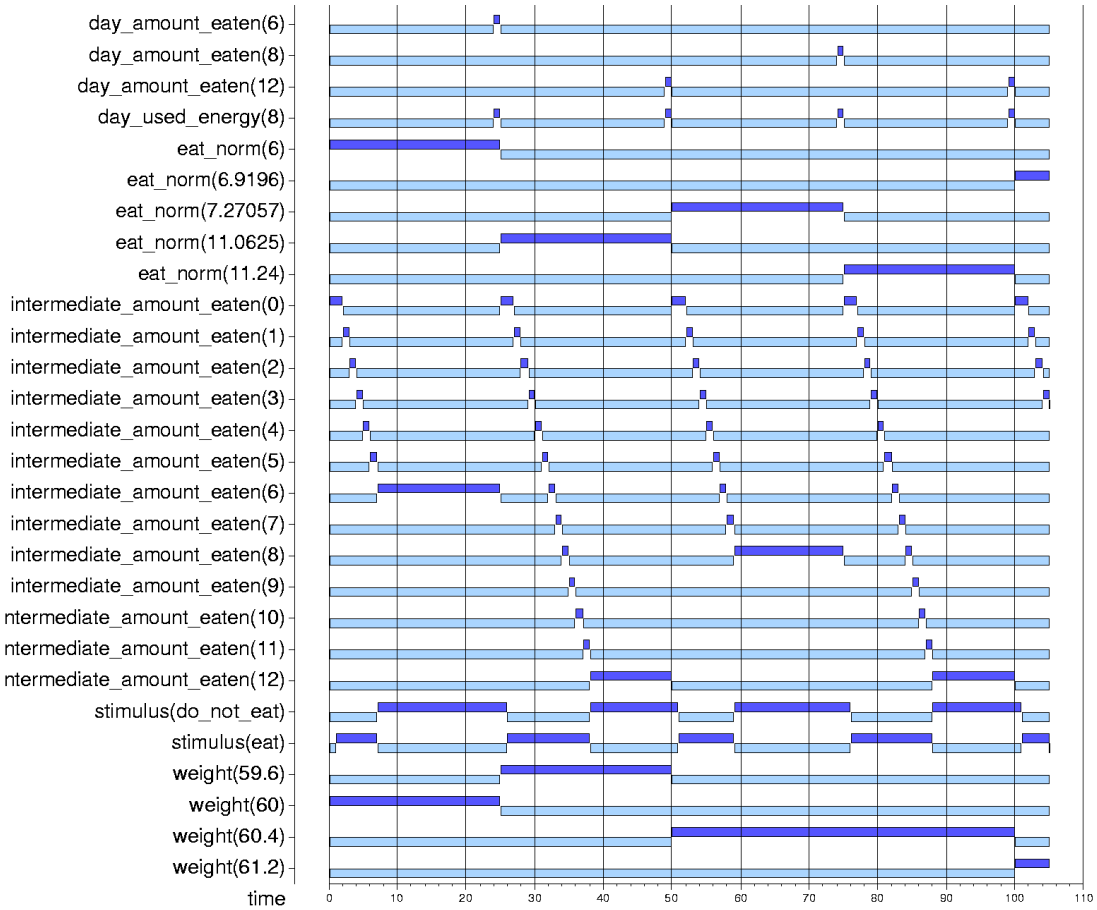


Figure 5. Simulation of the pattern of a person with bulimia

5. Analysis of Dynamic Properties of the System

Complex dynamic processes can be described at different aggregation levels, varying from the *local* level of (generating) basic mechanisms to the level of (emerging) *global* dynamic properties of a process as a whole. To analyse how such global dynamic

properties relate to local properties it is useful to distinguish *intermediate* properties. Moreover, some other (*environmental*) properties may be needed that relate the considered process to other processes that are not modelled and considered as external environment. In this section, the different types of non-local dynamic properties of the system are identified. For the relationships between the properties see also Figure 6.

For the adaptive dynamical system, the amount of used energy is an exogenous variable, i.e., this comes from the *environment*. To be able to do analysis, it is convenient to consider certain simplifying assumptions on the environment. For example, to study limit behaviour, a suitable assumption is that from a certain point of time no changes occur in the used energy (EP2), or to study how the system behaves under one change, a suitable assumption is that only one change occurs in the environment (EP1). The latter type of environment may be used, for example, to study transitions occurring in subjects of around 35 years old, when the metabolism becomes slower, and hence the day amount of used energy will become lower. For each of the properties, first an informal description is given, and next the formal description that has been used for the automated checking software; see Discussion.

EP1(t1, t2, E1, E2) (Transition from one used energy E1 to another used energy E2)
Property EP1 expresses that first the day amount of used energy is constant at value E1, and next it is constant at (another) value E2. Formalisation:
For all $t < t1$ state($\mathcal{T}$, t) $\models$ day_used_energy(E1)
& for all $t \geq t2$ state($\mathcal{T}$, t) $\models$ day_used_energy(E2)

EP2(t, E) (Constant amount of used energy E from time t)
Property EP2 expresses that from a certain time point t the day amount of used energy is constant E. Formalisation:
For all $t' \geq t$ state($\mathcal{T}$, t') $\models$ day_used_energy(E)

Global properties (GP) are dynamic properties of the process as a whole.

GP1(W, m) (Stable weight W, margin m, e.g., 2%)
Property GP1 expresses that fluctuations in weight are limited to a relative m-interval of weight W. Formalisation:
For all t [state($\mathcal{T}$, t) $\models$ weight(W1) $\Rightarrow$ $-m \leq (W1 - W)/W \leq m$]

GP2(t1, t2, E1, E2, W, m) (Conditional constant weight W with margin m)
Property GP2 states that GP1 holds in environments in which only one change occurs in the day amount of used energy. Formalisation:
EP1(t1, t2, E1, E2) $\Rightarrow$ GP1(W, m)

GP3(t, E, d, e) (Adaptation of day amount eaten)
Property GP3 expresses that if the day amount of used energy is constant E after a time point t, then the day amount of food eaten will be in a relative d-interval of E. Formalisation:
For all t EP2(t, E) $\Rightarrow$ $\exists t'$ $t \leq t' \leq t + e$ & state($\mathcal{T}$, t') $\models$ time(24) &
$\forall E1$[state($\mathcal{T}$, t') $\models$ day_amount_eaten(E1) $\Rightarrow$ $-d \leq (E1 - E)/E \leq d$]

Intermediate properties are dynamic properties, normally fulfilled by parts of the dynamical system such that together they entail the global properties.

IP1(t, E, d, e) (Eat norm is adapting to used energy)
Intermediate property IP1 expresses that, if the day amount of used energy is constant after time point t, than, after some time the eat norm will be in a relative d-interval of E. Formalisation:
For all t EP2(t, E) $\Rightarrow$ $\exists t'$ $t \leq t' \leq t + e$ & state($\mathcal{T}$, t') $\models$ time(24) &
[state($\mathcal{T}$, t') $\models$ eat_norm(N) $\Rightarrow$ $-d \leq (N - E)/E \leq d$]

IP2 (Eat stimuli)
Intermediate property IP2 expresses how the eat norm N and the amount of food eaten together determine whether or not an eat stimulus occurs. It is just the conjunction of LP1 and LP2. Formalisation:
LP1 & LP2

IP3 (Day eating accumulation)
Intermediate property IP3 expresses how the day amount of eaten food is generated by following the eat stimuli during the day. Formalisation:
LP3 & LP4 & LP5

6. Interlevel Relationships Between Properties

The dynamic properties as identified in the section above describe the process at different levels of aggregation. The global properties describe the highest aggregation level: of the process as a whole. The local properties presented earlier describe the process at the lowest level of aggregation: the specific basic mechanisms. These properties are logically related in the sense that if a trace satisfies all local properties, then it also satisfies the global properties. To analyse these logical relationships between properties at different aggregation levels more systematically, properties at an intermediate aggregation level have been defined: the intermediate properties. Thus a set of properties at different aggregation levels was obtained that forms a connected set of properties with the following interlevel relationships:

EP1(t1, t2, E1, E2) & GP2(W, m)	$\Rightarrow$	GP1(W, m)
GP3(d, e) & LP6	$\Rightarrow$	GP2(W, m)
IP1(d, e) & IP2 & IP3	$\Rightarrow$	GP3(d, e)
LP7	$\Rightarrow$	IP1(d)
LP1 & LP2	$\Rightarrow$	IP2
LP3 & LP4 & LP5	$\Rightarrow$	IP3

The interlevel relationships are depicted by an AND-tree in Figure 6. Here a property at a parent node is implied by the conjunction of the properties at its children nodes.

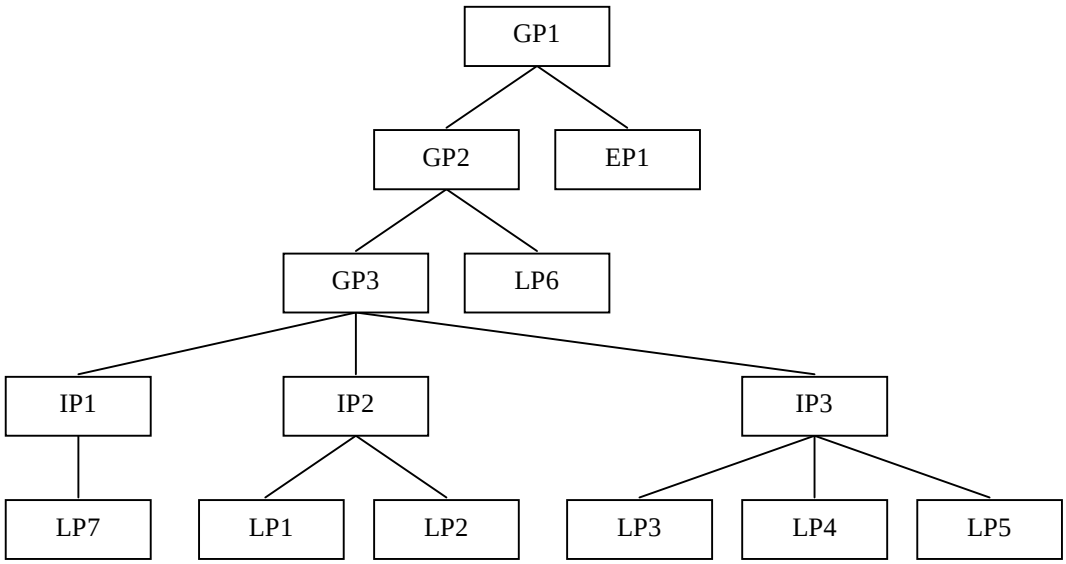


Figure 6. Interlevel relations between the dynamic properties

7. Diagnostics Based on Failing Analysis

The interlevel relations as depicted in Figure 6 provide a formalisation of a basis for a form of diagnostic reasoning that is sometimes applied in therapy practice. This reasoning runs as follows. Suppose the top level property GP1 fails (*e.g., non-stable weight).* Then due to the logical interlevel relations, one level lower in the tree either EP1 fails (*e.g., strongly fluctuating metabolism*) or GP2 fails. Suppose GP2 fails. Then one level lower either LP6 fails (*e.g., insufficient food uptake by digestion*) or GP3 fails. Suppose GP3 fails. Then either IP2 fails (*e.g., no effect of eatnorm on eating*) or IP3 fails (*e.g., eating no adequate food in the sense of energy-content*) or IP1 fails. Suppose IP1 fails. Then LP7 fails (*e.g., no adequate adaptation mechanism of eat norm to energy use*). Subsequently the type of failure of LP7 can be identified depending on whether weight is systematically too low or decreasing (first phase anorexia), too high, or increasing (first phase obesitas) or fluctuating (first phase bulimia).

8. Discussion

Two software environments have been developed to support the research reported here. First a simulation environment has been used to generate simulation traces as shown. Second, checking software has been used that takes traces and formally specified properties and checks whether a property holds for a trace.

	trace 1	**trace 2**	**trace 3**	**trace 4**	**trace 5**
EP1	+	+	+	+	+
EP2	+	+	+	+	+
GP1	+	-	-	-	-
GP2	+	-	-	-	-
GP3	+	-	-	+	-
IP1	+	-	-	+	+
IP2	+	+	+	+	+
IP3	+	+	+	+	-
LP1	+	+	+	+	+
LP2	+	+	+	+	+
LP3	+	+	+	+	-
LP4	+	+	+	+	+
LP5	+	+	+	+	+
LP6	+	+	+	-	+
LP7	+	-	-	+	+

Table 1. Results of checking properties against traces

The results for checking the properties on a number of these traces are as depicted in Table 1. The parameters used were as follows: W = 60, E = 8, m = 0.02, d = 0.1 and e = 24. Here the first three traces are those depicted in Figure 2, 4 and 5 respectively (normal, anorexia and bulimia). In traces 2 and 3 the adaptation mechanism is malfunctioning (LP7 is the cause of the problems). Trace 4 shows a pattern in which the eating regulation in principle functions well but there is insufficient food uptake by digestion (LP6 is the cause of the problems), whereas trace 5 shows a pattern in which the response on the eat

stimulus is eating food without energetic value (LP3 is the cause of the problems). Notice that indeed for all these traces the interlevel relations of Figure 6 hold.

In comparison to Temporal Logic (Barringer et al., 1996) our simulation approach has possibilities to incorporate (real or integer) numbers in state properties, and in the timing parameters e, f, g, h. Furthermore, TTL has more expressive power than temporal logic. For example, explicit reference can be made to (real) time, and variables can be used. Moreover, reference can be made to different developments of processes over time; thus statements such as 'exercise improves skill', which require comparison of different histories, can be formalised.

In comparison to rule-based approaches such as described by Holland (1995) and Rosenbloom, Laird and Newell (1993), our leads to format is more declarative in a temporal sense: in a built-in manner the simulation processes are explicitly related to (and have their semantics in) the (real) time dimension, and that relationship to time does not depend on the computational processes in an implicit manner, as in rule processing is usual. Furthermore, in our approach a format is available to express more complex, non-executable dynamic properties in our language TTL, and analysis methods for these dynamic properties at different aggregation levels are available as described above.

The high-level model integrates both medical and psychological aspects of the process, and has proven its value by predicting and explaining many of the patterns observed in psychotherapy practice. As one example, the development of obesitas after the age of 35 year can be explained as a lack of adaptive properties of the system with respect to decreased metabolism level. A more detailed model based on a set of differential equations for more detailed physiological processes is hard to obtain due to the lack of detailed knowledge (and parameter values) at the physiological level. Furthermore, even if such a model could be constructed, it probably would be so complex that it is hard to handle for simulation and analysis. Moreover, such mathematical techniques are not compatible with the type of reasoning within psychotherapy practice.

Further work is underway to address further phases of eating regulation disorders, especially phases when the subject's more conscious cognitive mechanisms to cope with such a disorder becomes more dominant. One of the aims is to show how, for cases of a malfunctioning system, the types of therapy described in (Delfos, 2002) can lead to a modified dynamical system in which eating regulation is functioning well.

References

Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M. (1996). *The Imperative Future: Principles of Executable Temporal Logic*, Research Studies Press Ltd. and John Wiley & Sons.

Beumont, P.J.V., et al. (1987). *Neural Basis of Appetite and Food Intake. Handbook of Eating Disorders,* New York: Elsevier Science Publishing Co.

Delfos, M.F. (2002). *Lost Figure: Treatment of Anorexia, Bulimia and Obesitas (in Dutch).* Swets and Zeitlinger Publishers, Lisse.

Garner, D.M. and P.E. Garfinkel, eds. (1985). *Handbook of Psychotherapy for Anorexia Nervosa and Bulimia,* New York: The Guilford Press.

Holland, J.H. (1995). *Hidden Order: How Adaptation Builds Complexity*. Addison-Wesley.

Jonker, C.M., and Treur, J. (2002). Analysis of the Dynamics of Reasoning Using Multiple Representations. In: W.D. Gray and C.D. Schunn (eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci 2002.* Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2002, pp. 512-517.

Jonker, C.M., Treur, J., and Wijngaards, W.C.A. (2003). A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. *Cognitive Systems Research Journal*, vol. 4(3), 2003, pp. 191-210.

Kupper, Z., and Hoffmann, H. (1996). A Boolean Approach to the Dynamics of Psychosis. In: W. Sulis and R. Combs (eds.), *Nonlinear Dynamics in Human Behaviour*. World Scientific, Singapore, pp. 296-315.

Levine, D.S., (1996). What Can Academic Psychology Contribute to Psychotherapy? *Psychline*, Vol. 1, No. 2, pp. 18-19, 1996.

Port, R.F., Gelder, T. van (eds.) (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, Mass.

Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

Rosenbloom, P.S., Laird, J.E., Newell, A. (eds.) (1993). *The SOAR Papers: Research on Integrated Intelligence*. MIT Press, Cambridge, MA.

Tschacher, W., Scheier, C., and Grawe, K., (1998). Order and Pattern Formation in Psychotherapy. *Nonlinear Dynamics, Psychology and Life Sciences*, vol. 2, pp. 195-215

Vink, T., Hinney, A., Van Elburg, A.A., Van Goozen, S.H., Sandkuijl, L.A., Sinke, R.J., Herpertz-Dahlman, B.M., Henebrand, J., Remschmidt, H., Van Engeland, H. & Adan, R.A. (2001). Association between an agouti-related protein gene polymorphism and anorexia nervosa. *Mol. Psychiatry*, 6 (3): 325-328

Warren, K., Sprott, J.C., and Hawkins, R.C., (2002). The Spirit Is Willing: Nonlinearity, Bifurcations, and Mental Control. *Nonlinear Dynamics, Psychology, and Life Sciences*, vol. 6, pp. 55-70.

PART V SINGLE VS. MULTIPLE AGENTS

CHAPTER 12

Simulation and Analysis of Shared Extended Mind

This chapter will appear as Bosse, T., Jonker, C.M., Schut, M.C., and Treur, J. (2005). Simulation and Analysis of Shared Extended Mind. *Simulation Journal: Transactions of the Society for Modeling and Simulation International*.

Part of this chapter appeared as Bosse, T., Jonker, C.M., Schut, M.C., and Treur, J. (2005). Simulation and Analysis of Shared Extended Mind. In: Davidsson, P., Gasser, L., Logan, B., and Takadama, K. (eds.), *Proceedings of the Joint Workshop on Multi-Agent and Multi-Agent-Based Simulation, MAMABS'04*. Lecture Notes in Artificial Intelligence, vol. 3415, Springer Verlag, pp. 248-264.

Simulation and Analysis of Shared Extended Mind

Tibor Bosse[1], Catholijn M. Jonker[1,2], Martijn C. Schut[1], and Jan Treur[1,3]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: {tbosse, jonker, schut, treur}@cs.vu.nl
URL: http://www.cs.vu.nl/~{tbosse, jonker, schut, treur}
[2] Universiteit Utrecht, Department of Philosophy,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Abstract. Some types of animals exploit patterns created in the environment as external mental states, thus obtaining an extension of their mind. In the case of social animals the creation and exploitation of such patterns can be shared, which supports a form of shared extended mind or collective intelligence. This paper explores this shared extended mind principle for social animals in more detail. The focus is on formal analysis and formalisation of the dynamic properties of the processes involved, both at the local level (the basic mechanisms) and the global level (the emerging properties of the whole), and their relationships. A case study in social ant behaviour in which shared extended mind plays an important role is used as illustration. For this case simulations are described based on specifications of local properties, and global properties are specified and verified.

1. Introduction

In [6], [7], [8], [10], [17], [19] it is described how behaviour is often not only supported by an internal mind in the sense of internal mental structures and cognitive processes, but also by processes based on patterns created in the external environment that serve as external mental structures. Examples of this pattern of behaviour are the use of 'to do lists' and 'lists of desiderata'. Having written these down externally (e.g., on paper, in your diary, in your organiser or computer) makes it unnecessary to have an internal memory about all the items. Thus internal mental processing can be kept less complex. The only thing to remember is where these lists are available. Other examples of the use of extended mind are doing mathematics or arithmetic, where external (symbolic, graphical, material) representations are used; e.g., [4].

Clark and Chalmers [8] point at the similarity between cognitive processes in the head and some processes involving the external world. This similarity can be used as an indication that these processes can be considered extended cognitive processes or extended mind:

> *'If, as we confront some task, a part of the world functions as a process which,* were it done in the head, *we would have no hesitation in recognizing as part of the cognitive process, then that part of the world* is *(so we claim) part of the cognitive process. Cognitive processes ain't (all) in the head!' [8], Section 2, p. 8. (…) 'Of course, one could always try to explain my action in terms of internal processes and a long series of "inputs" and "actions", but this explanation would be needlessly complex. If an isomorphic process were going on in the head, we would feel no urge to characterize it in this cumbersome way.' [8], Section 3, p.10.*

We will call this criterion the 'isomorphism' criterion. As the patterns in the external world have to be created and sensed, interaction with the external world will be more intensive, compared to the case where internal mental states are created and exploited.

Especially in the case of social animals external mental states created by one individual can be exploited by another individual, or, more generally, the creation, maintenance, and exploitation of external mental states are activities in which a number of individuals can participate (for example, presenting slides on a paper with multiple authors to an audience). Further examples can be found everywhere, varying from roads and traffic signs to books or other media, and to many other kinds of cultural achievements. In this multi-agent case the extended mind principle serves as a way to build a form of social or collective intelligence, that goes beyond (and may even not require) social intelligence based on direct one-to-one communication. In such cases the external mental states cross, and in a sense break up, the borders between (the minds of) the individuals and become *shared extended mental states*.

An interesting and currently often studied example of collective intelligence is the intelligence shown by ant colonies [2], [9], [11]. Indeed, in this case the external world is exploited as an extended mind by using pheromones. While they walk, ants drop pheromones on the ground. The same or other ants sense these pheromones and follow the route in the direction of the strongest concentration. Because pheromones evaporate, such routes may vary over time. This context is chosen in this paper to illustrate the shared extended mind principle.

The main contribution of the current paper is a detailed analysis of this shared extended mind principle, and a formalisation of its dynamics. The principle is illustrated by a case study of social behaviour based on shared extended mind (a simple ant colony). The analysis of this case study comprises multi-agent simulation based on identified local dynamic properties, identification of dynamic properties for the overall process, and verification of these dynamic properties. The shared extended mind principle and its formalisation as introduced in this paper allow one to perform simulation and explanation of behaviour on a more abstract level: in terms of mental states instead of the physical materialisation. This provides a simpler, more abstract and perhaps more understandable and elegant interpretation of the simulation models, than based on the physical counterparts. This is made possible by interpreting the external world states involved according to a new ontology. Considering part of the external world as extended mind allows one to give another interpretation to external physical processes and states. Physical state properties such as 'pheromone is present at d' can be interpreted (and even renamed) as, for example, 'it is believed that d is the direction home'. In fact this double interpretation still gives two possibilities: for empirical data the physical interpretation can be chosen, whereas for modelling the other, mental interpretation can be kept in mind.

More specifically, in this paper Section 2 is a brief introduction of the basic concepts used in the modelling approach and formalisation. It introduces two modelling languages, one (the *Leads To* language) used for simulation, and one (the *Temporal Trace Language* TTL) for more complex properties that can be used in analysis. For the former language a software environment for simulation has been developed, for the latter language a software environment has been developed that enables automatic checking of specified properties against given traces.

In Section 3 the extended mind principle is formalised by an isomorphic mapping between a cognitive process using external mental states and a similar process based on internal mental states. This mapping formalises how the processes of the agent in interaction with the world indeed can be interpreted as (comparable to) an agent with internal mental processes, thus formalising the 'were it done in the head' criterion phrased in the citation from Clark and Chalmers [8] given above. In Section 4 a simulation model is presented for the ant case study. This simulation model is specified using local properties: temporal rules that express in a local manner the basic mechanisms of the case.

These rules are specified and formalised in the *leads to* language introduced in Section 2, and are therefore directly executable in the software environment that has been developed. Some of the simulation outcomes are included in Section 4. Whereas Section 4 has a local perspective on the basic mechanisms, Section 5 takes the global perspective of emergent properties of the multi-agent process as a whole. A number of relevant global dynamic properties are identified and formalised in the language TTL. It is discussed how these global dynamic properties have been checked against simulation traces. Moreover, some of the logical relationships between them are discussed. Section 6 is a discussion of the results.

2. State Properties and Dynamic Properties

Dynamics will be described in the next section as evolution of states over time. The notion of state as used here is characterised on the basis of an ontology defining a set of physical and/or mental (state) properties that do or do not hold at a certain point in time. States can be taken as global states, but also more local perspectives, based on a subset of the overall ontology, can be expressed in a state, for example an internal agent state. as an example, the internal state property 'the agent A has pain', or the external world state properties 'it is raining' and 'the environmental temperature is 7° C', may be expressed in terms of different ontologies. To formalise state property descriptions, an ontology is specified as a finite set of sorts, constants within these sorts, and relations and functions over these sorts. The example properties mentioned above then can be defined by nullary predicates (or proposition symbols) such as itsraining, or by using n-ary predicates (with $n \geq 1$) like has_pain(A) and has_temperature(environment, 7). For a given ontology Ont, the propositional language signature consisting of all *state ground atoms* (or *atomic state properties*) based on Ont is denoted by APROP(Ont). The *state properties* based on a certain ontology Ont are formalised by the propositions that can be made (using conjunction, negation, disjunction, implication) from the ground atoms. A *state* S is an indication of which atomic state properties are true and which are false, i.e., a mapping S: APROP(Ont) $\rightarrow$ {true, false}.

To describe the internal and external dynamics of the agent, explicit reference is made to time. Dynamic properties can be formulated that relate a state at one point in time to a state at another point in time. A simple example is the following informally stated dynamic property for belief creation based on observation:

'if the agent observes at t1 that it is raining, then the agent will believe that it is raining'.

To express such dynamic properties, and other, more sophisticated ones, the Temporal Trace Language TTL is used; cf. [14]. In this language, explicit references can be made to time points and traces. Here a *trace* or *trajectory* over an ontology Ont is a time-indexed sequence of states over Ont. The sorted predicate logic temporal trace language TTL is built on atoms referring to, e.g., traces, time and state properties. For example, 'in trace γ at time t property p holds' is formalised by state(γ, t) |= p. Likewise, 'in trace γ at time t property p does not hold' is formalised by state(γ, t) |$\neq$ p. Here |= is a predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus. Dynamic properties are expressed by temporal statements built using the usual logical connectives and quantification (for example, over traces, time and state properties). For example, consider the following dynamic property:

'in any trace γ, if at any point in time t1 the agent A observes that it is raining,
then there exists a time point t2 after t1 such that at t2 in the trace the agent A believes that it is raining'.

In formalised TTL form it looks as follows:

$\forall t1$ [state(γ, t1) |= observes(A, itsraining) $\Rightarrow$
$\exists t2 \geq t1$ state(γ, t2) |= belief(A, itsraining)]

Language abstractions by introducing new (definable) predicates for complex expressions are possible and supported.

In order to specify simulation models, a simpler temporal language has been developed, based on TTL. This language (the *leads to* language) enables one to model direct temporal dependencies between two state properties in successive states. This executable format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically. Moreover, the language offers the possibility to express both qualitative and quantitative aspects of a process to be simulated. Therefore, it combines the advantages of logic-oriented approaches such as [1] and [13] with those of mathematical approaches like [20] in the context of simulation modeling and analysis [18]. For a more precise definition of the *leads to* format, see [3].

3. Explanation and the Isomorphism Principle

In Section 1 the isomorphism principle was introduced, based on the apparent similarity between cognitive processes in the head and some processes involving the external world. For an illustration of this principle see Figure 1 and 2. In these figures, the circles denote state properties, the arrows denote dynamic properties, and the dotted box indicates the borders of the agent.

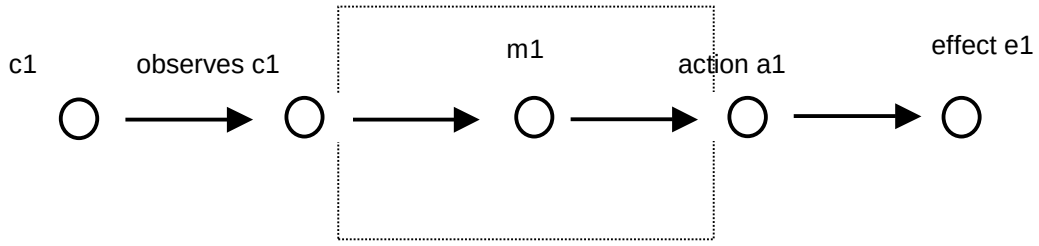


Figure 1. Behaviour based on an internal mental state

Figure 1 depicts a simple case of an agent with behaviour based on an internal mental state property m1, whereas Figure 2 depicts another agent with the same behaviour based on an external mental state property m2. In both cases, the internal (m1) or external (m2) state property acts as a mediator in the trajectory between input (c1) and output (e1). Thus, in a way both m1 and m2 can be considered an agent's belief. Note that the internal processing of the agent in Figure 2 is chosen as simple as possible: stimulus response. Hence, this agent is assumed not to have any internal states. This is in line with the ideas of Clark and Chalmers, who claim that the explanation of cognitive processes should be as simple as possible [8]. However, the interaction between this agent and the external

world is a bit more complex than in Figure 1: one extra action is needed to create the external mental state m2, and one additional observation is needed to observe it.

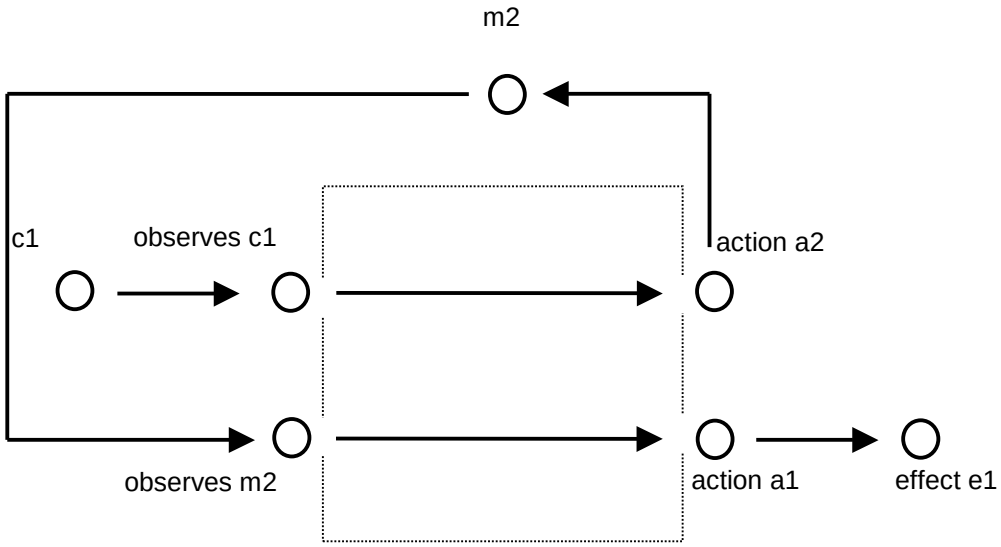


Figure 2. Behaviour based on an external mental state

To make the similarity between the two different cognitive processes more precise, the following mapping from the upper graph into the lower graph can be made (see Figure 3):

c1	→	c1
observes c1	→	observes c1
m1	→	m2
action a1	→	action a1
effect e1	→	effect e1

This mapping, indicated by the vertical dotted arrows in Figure 3, preserves the temporal (*leads to*) relationships (the solid arrows) and provides an (isomorphic in the mathematical sense) embedding of a cognitive process based on internal mind into a cognitive process based on extended mind. Remember the quotes from [8], cited in the Introduction. Clark and Chalmers [8] use the isomorphism to a process 'in the head' as one of the criteria to consider external and interaction processes as cognitive, or mind processes.

This 'isomorphism' criterion is formalised in Figure 3 for a simple example of such an isomorphism. Note that the process from m1 to action a1, modelled as one step in the internal case, is mapped onto a process from m2 via observes m2 to action a1, in the external case modelled as a two-step process. So the isomorphism is an embedding in one direction, not a bidirectional isomorphism, simply because the observation state for m2 (and the same for the action a2) has no counterpart in the internal case. For a more detailed treatment of the isomorphism, and an extension of the mapping to formally defined dynamic properties, see [5].

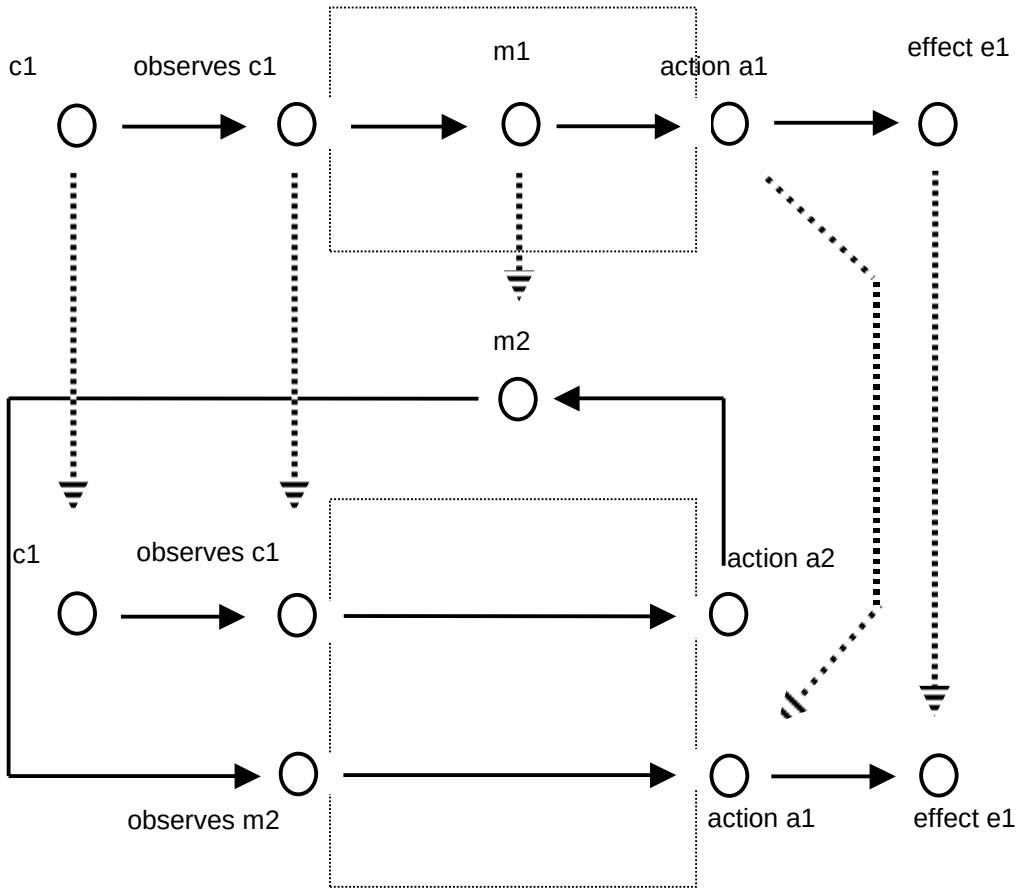


Figure 3. Internal and external mental states and their isomorphism relationship

Behaviour often is explained by considering the basic underlying causal relations or mechanisms. Such basic mechanisms can be formally modelled by *leads to* relations. The isomorphism principle and its formalisation as depicted in Figure 3 allows one to replace an explanation of behaviour in terms of basic mechanisms involving frequent interactions (observations and actions) with the external world, by an explanation that leaves out these interactions and bases itself directly on the mental states. This explanation is simpler, more abstract and perhaps more elegant, than the more complicated 'cumbersome' explanation based on the interactions. This is made possible by introducing a new ontology for the external world states involved. Considering part of the external world as extended mind allows one to give another interpretation to external physical processes and states. Physical state properties such as 'pheromone is present at d' are renamed as, for example, 'it is believed that d is the direction home'. Why would one introduce extra language to refer to the same fact in the world? Given the literature on reduction, where often it is claimed that mental state properties can be and actually should be replaced by their physical realisers, at first sight such an opposite move may seem a bit surprising. For example, Kim [16] (pp. 214-216) claims that ontological simplification is one of the reasons to reduce mental state properties to physical state properties. In the extended mind case at hand the converse takes place; the question is what is the advantage of this "ontological complication". A number of arguments in support of this can be given. Clark and Chalmers [8] claim that this allows application of other types of explanation and other methods of scientific investigation:

'… we allow a more natural explanation of all sorts of actions. (…) in seeing cognition as extended one is not merely making a terminological decision; it makes a significant difference to the methodology of scientific investigation. In effect, explanatory methods that might once have been thought appropriate only for the analysis of "inner" processes are now being adapted for the study of the outer, and there is promise that our understanding of cognition will become richer for it.' [8], Section 3, p. 10.

In [15] it is explained in some detail, and illustrated by examples, why in various cases in other areas (such as Computer Science) such an antireductionist strategy often pays off. Advantages in terms of insight, transparency and generality include: additional higher-level ontologies can improve understanding as they may allow simplification of the picture by abstracting from lower-level details; more insight is gained from a conceptually higher-level perspective; analysis of more complex processes is possible; the same concepts have a wider scope of application, thus obtaining unification. For more details and support for this antireductionist argument, see [15].

4. A Simulation Model of Shared Extended Mind

Dynamic properties can be specified at different aggregation levels, varying from (local) dynamic properties for the basic mechanisms and (global) properties of a process as a whole. This section introduces the local dynamic properties for the basic mechanisms; they are used to specify a simulation model. The world in which the ants live is described by a labeled graph as depicted in Figure 4.

Locations are indicated by A, B,…, and edges by E1, E2,… The ants move from location to location via edges; while passing an edge, pheromones are dropped. The objective of the ants is to find food and bring this back to their nest. In this example there is only one nest (at location A) and one food source (at location F).

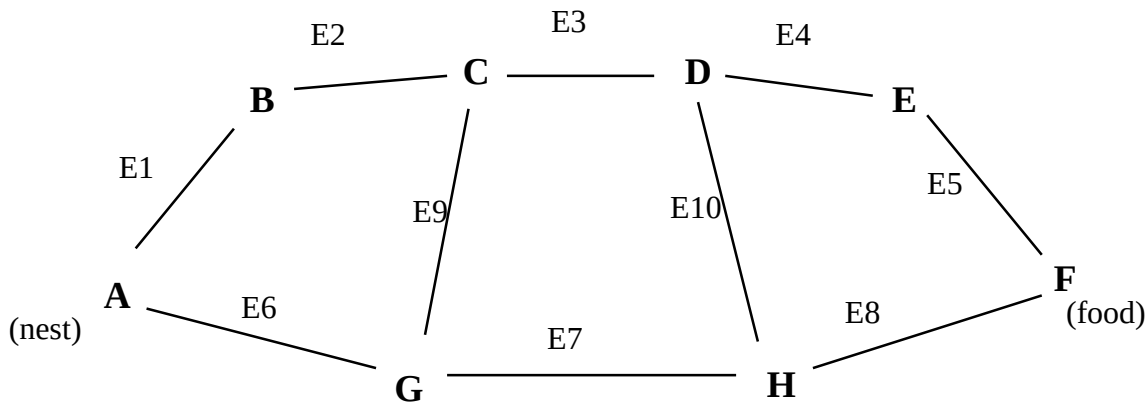


Figure 4. An ants world

The example concerns multiple agents (the ants), each of which has input (to observe) and output (for moving and dropping pheromones) states, and a physical body which is at certain positions over time, but no internal mental state properties (they are assumed to act purely by stimulus-response behaviour). An overview of the formalisation of the state properties of this single agent conceptualisation is shown in Table 1. In these local properties, a is a variable that stands for ant, e for edge, i for pheromone level, l for location, and n for number of neighbour locations. Note that in some of the state properties the direction of an ant is incorporated (e.g., ant a is at location l coming from e, ant a is at edge e to l2 coming from location l1). This direction is meant to relate to the

orientation of the ant's body in space, which is a genuine state property; but for convenience this is expressed by referring to the past or future states involved.

	body positions in world:
pheromone level at edge e is i	pheromones_at(e, i)
ant a is at location l coming from e	is_at_location_from(a, l, e)
ant a is at edge e to l2 coming from location l1	is_at_edge_from_to(a, e, l1, l2)
ant a is carrying food	is_carrying_food(a)
	world state properties:
edge e connects location l1 and l2	connected_to_via(l1, l2, e)
location 1 is the nest location	nest_location(l)
location 1 is the food location	food_location(l)
location l has n neighbours	neighbours(l, n)
edge e is most attractive for ant a coming from location l	attractive_direction_at(a, l, e)
	input state properties:
ant a observes that it is at location l coming from edge e	observes(a, is_at_location_from(l, e))
ant a observes that it is at edge e to l2 coming from location l1	observes(a, is_at_edge_from_to(e, l1, l2))
ant a observes that edge e has pheromone level i	observes(a, pheromones_at(e, i))
	output state properties:
ant a initiates action to go to edge e to l2 coming from location l1	to_be_performed(a, go_to_edge_from_to(e, l1, l2))
ant a initiates action to go to location l coming from edge e	to_be_performed(a, go_to_location_from(l, e))
ant a initiates action to drop pheromones at edge e coming from location l	to_be_performed(a, drop_pheromones_at_edge_from(e, l))
ant a initiates action to pick up food	to_be_performed(a, pick_up_food)
ant a initiates action to drop food	to_be_performed(a, drop_food)

Table 1. Formalisation of state properties

Below, the local dynamic properties are shown that were used to model the example. On the left, the dynamic properties are given in formal *leads to* format. In each dynamic property, the values 0, 0, 1, 1 were chosen for the timing parameters e, f, g, h (see Section 2). For simplicity, these parameters were left out in the description below. On the right, for each dynamic property an informal description is provided.

LP1 (Initialisation of Pheromones) start $\twoheadrightarrow$ pheromones_at(E1, 0.0) and pheromones_at(E2, 0.0) and pheromones_at(E3, 0.0) and pheromones_at(E4, 0.0) and pheromones_at(E5, 0.0) and pheromones_at(E6, 0.0) and pheromones_at(E7, 0.0) and pheromones_at(E8, 0.0) and pheromones_at(E9, 0.0) and pheromones_at(E10, 0.0)	At the start of the simulation, at all locations there are 0 pheromones.
LP2 (Initialisation of Ants) start $\twoheadrightarrow$ is_at_location_from(ant1, A, init) and is_at_location_from(ant2, A, init) and is_at_location_from(ant3, A, init)	At the start of the simulation, all ants (in this case, Ant 1, 2 and 3) are at location A. The exact time point an ant is added to the simulation can be specified manually.
LP3a (Initialisation of World) start $\twoheadrightarrow$ connected_to_via(A, B, l1) and … and connected_to_via(D, H, l10)	This property expresses which locations are connected to each other, and via which edges they are connected.
LP3b (Initialisation of World) start $\twoheadrightarrow$ neighbours(A, 2) and … and neighbours(H, 3)	This property expresses for each location how many neighbours it has.
LP4 (Initialisation of Attractive Directions) start $\twoheadrightarrow$ attractive_direction_at(ant1, A, E1) and … and attractive_direction_at(ant3, E, E5)	This property expresses for each ant and location, which edge is most attractive for the ant if it arrives at that location. This criterion is used only when there are edges with equal pheromone levels[6].
LP5a (Selection of Edge) observes(a, is_at_location_from(A, e0)) and attractive_direction_at(a, A, e1) and connected_to_via(A, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(A, l2, e2) and observes(a, pheromones_at(e2, i2)) and e1 $\neq$ e2 and i1 = i2 $\twoheadrightarrow$ to_be_performed(a, go_to_edge_from_to(e1, A, l1))	If an ant observes that it is at location A, and both edges connected to location A have the same number of pheromones, then the ant goes to its attractive direction.
LP5b (Selection of Edge) observes(a, is_at_location_from(A, e0)) and connected_to_via(A, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(A, l2, e2) and observes(a, pheromones_at(e2, i2)) and i1 > i2 $\twoheadrightarrow$ to_be_performed(a, go_to_edge_from_to(e1, A, l1))	If an ant observes that it is at location A, and one edge connected to location A has the highest number of pheromones, then the ant goes to that edge.
LP5c (Selection of Edge) observes(a, is_at_location_from(F, e0)) and connected_to_via(F, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(F, l2, e2) and observes(a, pheromones_at(e2, i2)) and i1 > i2 $\twoheadrightarrow$ to_be_performed(a, go_to_edge_from_to(e1, F, l1))	If an ant observes that it is at location F, and one edge connected to location F has the highest number of pheromones, then the ant goes to that edge.
LP5d (Selection of Edge) observes(a, is_at_location_from(l, e0)) and neighbours(l, 2) and connected_to_via(l, l1, e1) and e0 $\neq$ e1 and l $\neq$ A and l $\neq$ F $\twoheadrightarrow$ to_be_performed(a, go_to_edge_from_to(e1, l, l1))	If an ant observes that it is at a location (which is not A or F) with 2 neighbours, then it continues in the direction it was travelling to.

[6] To obtain interesting simulation traces, different attractive directions were assigned to different ants. However, another possibility (that is supported by the software) is to assign attractive directions at random.

LP5e (Selection of Edge) observes(a, is_at_location_from(l, e0)) and attractive_direction_at(a, l, e1) and neighbours(l, 3) and connected_to_via(l, l1, e1) and observes(a, pheromones_at(e1, 0.0)) and connected_to_via(l, l2, e2) and observes(a, pheromones_at(e2, 0.0)) and e0 ≠ e1 and e0 ≠ e2 and e1 ≠ e2 ↠ to_be_performed(a, go_to_edge_from_to(e1, l, l1))	If an ant observes that it is at a location with three neighbours, and all edges connected to that location have the same number of pheromones, then the ant goes to its attractive direction.
LP5f (Selection of Edge) observes(a, is_at_location_from(l, e0)) and neighbours(l, 3) and connected_to_via(l, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(l, l2, e2) and observes(a, pheromones_at(e2, i2)) and e0 ≠ e1 and e0 ≠ e2 and e1 ≠ e2 and i1 > i2 ↠ to_be_performed(a, go_to_edge_from_to(e1, l1))	If an ant observes that it is at a location with three neighbours, and one edge connected to that location has the highest number of pheromones, then the ant goes to that edge.
LP6 (Arrival at Edge) to_be_performed(a, go_to_edge_from_to(e, l, l1)) ↠ is_at_edge_from_to(a, e, l, l1)	If an ant goes to an edge e from a location l to a location l1, then later the ant will be at this edge e.
LP7 (Observation of Edge) is_at_edge_from_to(a, e, l, l1) ↠ observes(a, is_at_edge_from_to(e, l, l1))	If an ant is at a certain edge e, going from a location l to a location l1, then it will observe this.
LP8 (Movement to Location) observes(a, is_at_edge_from_to(e, l, l1)) ↠ to_be_performed(a, go_to_location_from(l1, e))	If an ant observes that it is at an edge e from a location l to a location l1, then it will go to location l1.
LP9 (Dropping of Pheromones) observes(a, is_at_edge_from_to(e, l, l1)) ↠ to_be_performed(a, drop_pheromones_at_edge_from(e, l))	If an ant observes that it is at an edge e from a location l to a location l1, then it will drop pheromones at this edge e.
LP10 (Arrival at Location) to_be_performed(a, go_to_location_from(l, e)) ↠ is_at_location_from(a, l, e)	If an ant goes to a location l from an edge e, then later it will be at this location l.
LP11 (Observation of Location) is_at_location_from(a, l, e) ↠ observes(a, is_at_location_from(l, e))	If an ant is at a certain location l, then it will observe this.
LP12 (Observation of Pheromones) is_at_location_from(a, l, e0) and connected_to_via(l, l1, e1) and pheromones_at(e1, i) ↠ observes(a, pheromones_at(e1, i))	If an ant is at a certain location l, then it will observe the number of pheromones present at all edges that are connected to location l.
LP13 (Increment of Pheromones) to_be_performed(a1, drop_pheromones_at_edge_from(e, l1)) and ∀l2 not to_be_performed(a2, drop_pheromones_at_edge_from(e, l2)) and ∀l3 not to_be_performed(a3, drop_pheromones_at_edge_from(e, l3)) and a1 ≠ a2 and a1 ≠ a3 and a2 ≠ a3 and pheromones_at(e, i) ↠ pheromones_at(e, i*decay+incr)	If an ant drops pheromones at edge e, and no other ants drop pheromones at this edge, then the new number of pheromones at e becomes i*decay+incr. Here, i is the old number of pheromones, decay is the decay factor, and incr is the amount of pheromones dropped.

LP14 (Collecting of Food) observes(a, is_at_location_from(l, e)) and food_location(l) $\rightarrow\!\!\!\!\rightarrow$ to_be_performed(a, pick_up_food)	If an ant observes that it is at location F (the food source), then it will pick up some food.
LP15 (Carrying of Food) to_be_performed(a, pick_up_food) $\rightarrow\!\!\!\!\rightarrow$ is_carrying_food(a)	If an ant picks up food, then as a result it will be carrying food.
LP16 (Dropping of Food) observes(a, is_at_location_from(l, e)) and nest_location(l) and is_carrying_food(a) $\rightarrow\!\!\!\!\rightarrow$ to_be_performed(a, drop_food)	If an ant is carrying food, and observes that it is at location A (the nest), then the ant will drop the food.
LP17 (Persistence of Food) is_carrying_food(a) and not to_be_performed(a, drop_food) $\rightarrow\!\!\!\!\rightarrow$ is_carrying_food(a)	As long as an ant that is carrying food does not drop the food, it will keep on carrying it.
LP18 (Decay of Pheromones) pheromones_at(e, i) and $\forall$a,l not to_be_performed(a, drop_pheromones_at_edge_from(e, l)) $\rightarrow\!\!\!\!\rightarrow$ pheromones_at(e, i*decay)	If the old amount of pheromones at an edge is i, and there is no ant dropping any pheromones at this edge, then the new amount of pheromones at e will be i*decay.

A special software environment has been created to enable the simulation of executable models. Based on an input consisting of dynamic properties in *leads to* format, the software environment generates simulation traces. Examples of such traces can be seen in Figure 5, 6 and 7. Time is on the horizontal axis, the state properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false. This trace is based on all local properties identified. Because of space limitations, in the example depicted in Figure 5 and 6, only three ants are involved. The trace in Figure 7 shows an example with two ants. However, similar experiments have been performed with populations of 50 and 100 ants. Since the abstract way of modelling used for the simulation is not computationally expensive, also these simulations can be performed relatively quickly. To be precise, they took 35 seconds (for 50 ants and 80 time steps), 70 seconds (100 ants, 80 time steps), 100 seconds (50 ants, 200 time steps), and 200 seconds (100 ants, 200 time steps), respectively.

Figures 5 and 6 are both parts from the same trace. Figure 5 shows the observations and locations of the ants; Figure 6 shows the performed actions of ant1 in more detail. As can be seen in Figure 5 and 6, there are two ants (ant1 and ant2) that start their search for food immediately (at time point 0), whereas ant3 comes into play a bit later, at time point 3. These time points were specified manually, see property LP2. When ant1 and ant2 start their search, none of the locations contain any pheromones yet, so basically they have a random choice where to go. In the current example, ant1 selects a rather long route to the food source (via locations A-B-C-D-E-F), whilst ant2 chooses a shorter route (A-G-H-F). Note that, in the current model, a fixed route preference (via the attractiveness predicate) has been assigned to each ant for the case there are no pheromones yet. After that, at time point 3, ant3 starts its search for food. At that moment, there are trails of pheromones leading to both locations B and G, but these trails contain exactly the same number of pheromones. Thus, ant3 also has a choice among location B and G, and chooses in this case to go to B. Meanwhile, at time point 18, ant2 has arrived at the food source (location F). Since it is the first to discover this location, the only present trail leading back to the nest, is its own trail. Thus ant2 will return home via its own trail. Next, when ant1 discovers the food source (at time point 31), it will notice that there is a trail leading back that is stronger than its own trail (since ant2 has already walked there twice: back and

forth, not too long ago). As a result, it will follow this trail and will keep following ant2 forever. Something similar holds for ant3. The first time that it reaches the food source, ant3 will still follow its own trail, but some time later (from time point 63) it will also follow the other two ants. To conclude, eventually the shortest of both routes is shown to remain, whilst the other route evaporates. Other simulations, in particular for small ant populations, show that it is important that the decay parameter of the pheromones is not too high. Otherwise, the trail leading to the nest has evaporated before the first ant has returned, and all ants get lost.

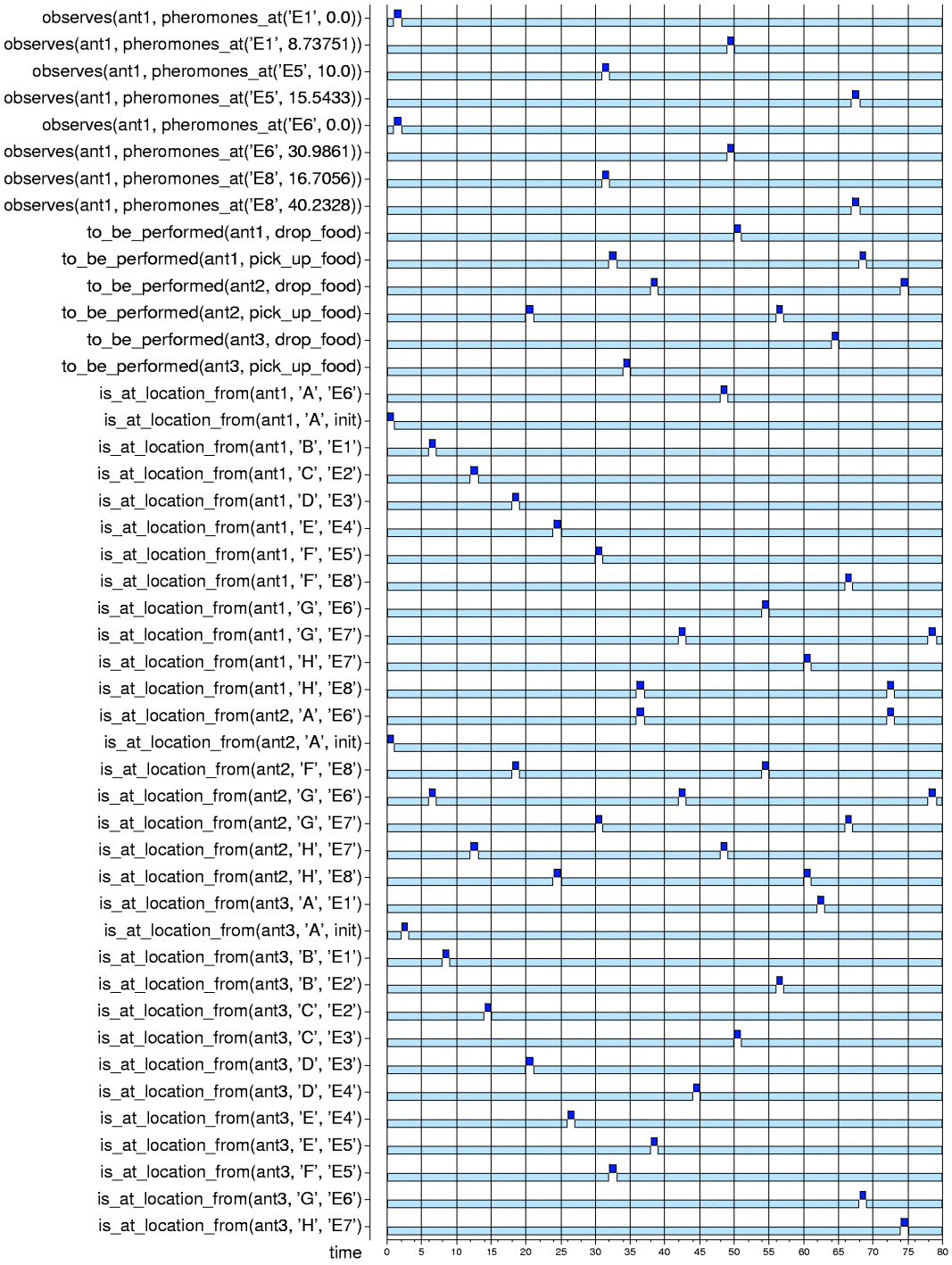


Figure 5. Simulation trace of the dynamics of the ants behaviour

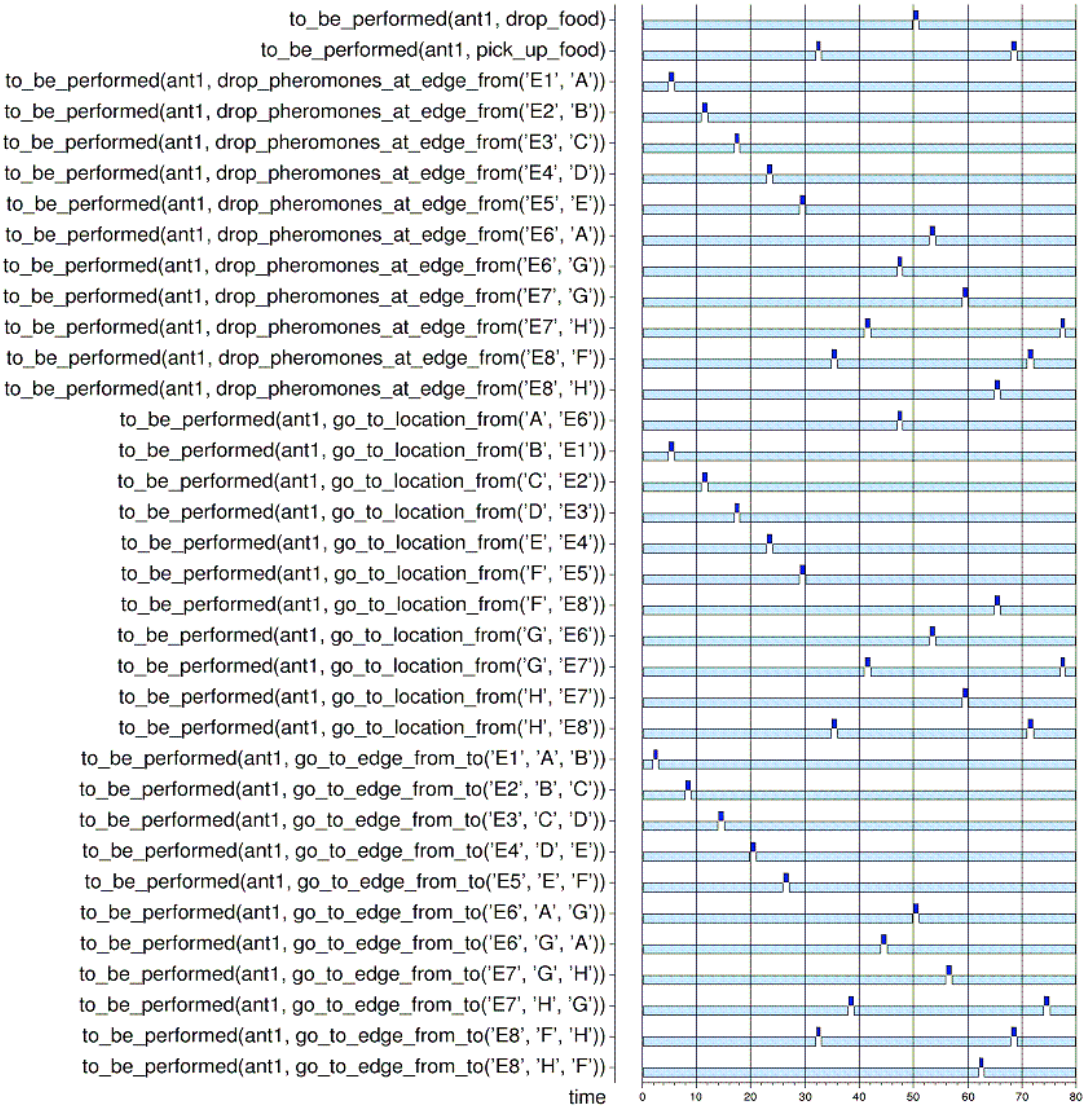


Figure 6. Simulation trace of the performed actions in the ant behaviour model

Figure 7 describes a different situation. In that figure, there is one ant (ant1) that starts its search departing from the food location and one ant (ant2) that starts slightly later (at time point 10), departing from the nest location. The first ant (ant1) takes the long way home (via locations F-E-D-C-B-A), while the second ant (ant2) immediately takes the short route (via locations A-G-H-F) to the food. Figure 7 shows that after some time, both ants follow the short route. Thus also for this example, we may conclude that eventually the shortest of both routes is shown to remain, whilst the other route evaporates.

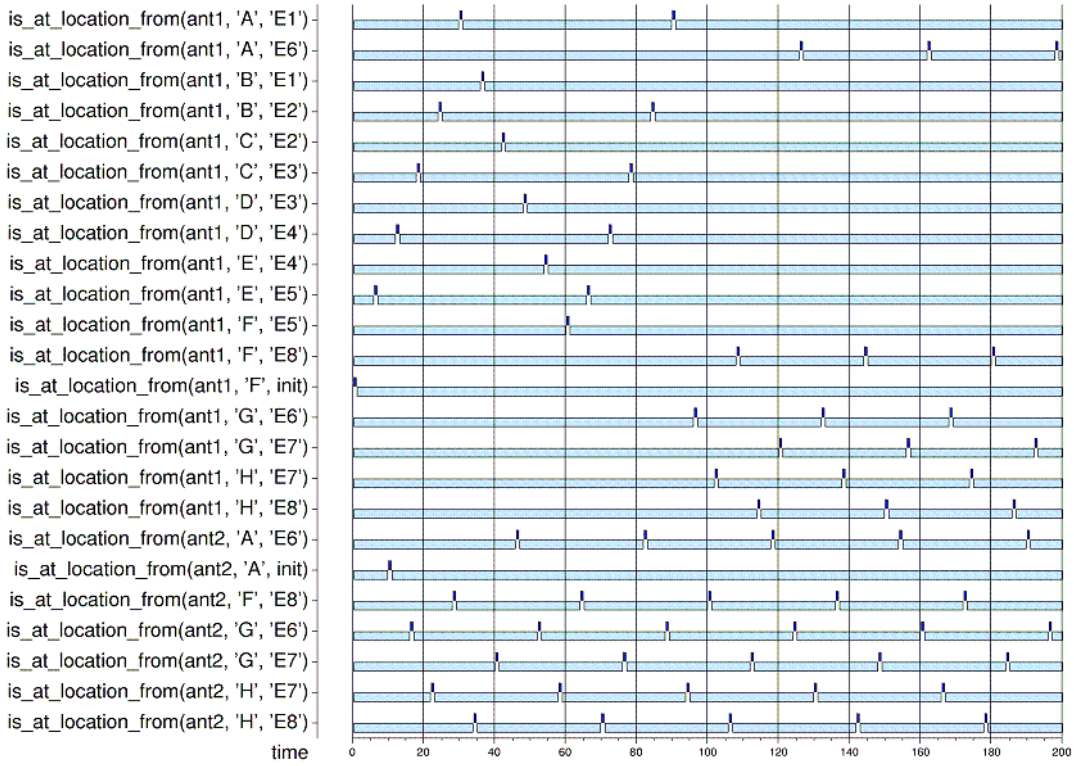


Figure 7. Simulation trace of the dynamics of the ants behaviour where ant1 departs from the food location, while ant2 depart slightly later from the nest location

5. Global Properties and Verification

In the previous section dynamic properties at the lowest aggregation level (the local dynamic properties) were addressed, and simulation based on these properties was discussed. The current section addresses dynamic properties of a global nature and their verification. Within these properties, γ is a variable that stands for an arbitrary trace. First a language abstraction is given:

food_delivered_by(γ, t, a) ≡ ∃l, e [state(γ,t) |= is_at_location_from(a, l, e)) &
state(γ,t) |= nest_location(l) & state(γ,t) |= to_be_performed(a, drop_food)]

GP1 Food Delivery Succesfulness
There is at least one ant that brings food back to the nest.
∃t ∃a: food_delivered_by(γ, t, a).

GP2 Multiple Delivery
Food is delivered by more than one ant
∃t1, t2 ∃a1, a2 [a1 ≠ a2 & food_delivered_by(γ, t1, a1) & food_delivered_by(γ, t2, a2)]

Another language abstraction is:
attractive_route_to(γ, a, x) ≡
∃l ∃e ∀t [state(γ, t) |= attractive_direction_at(a, l, e) & state(γ, t) |= connected_to_via(l, x, e)]
I.e., the attractive route of ant a (in case of equal pheromone levels) passes through location x.

Yet another language abstraction is:
reaches_end_attractive_route(γ, t, a) ≡
∃l, e [state(γ, t) |= is_at_location_from(a, l, e) &
attractive_route_to(γ, a, l) & ∀e' state(γ, t) |≠ attractive_direction_at(a, l, e')]

GP3 Reaching End of Attractive Route
Ants reach the end of their attractive route.
∀a ∃t reaches_end_attractive_route(γ, t, a)

GP4 Returning To Nest
Ants get back to the nest from the end of their attractive routes.
∀a ∀t1 ∃e, t2 > t1 ∃l [reaches_end_attractive_route(γ, t1, a) ⇒
state(γ, t2) |= is_at_location_from(a, l, e) & state(γ, t2) |= nest_location(l)]

GP5 From Food To Nest
Ants get back to the nest from locations of food.
∀a, e ∀t1 ∃t2 > t1 ∃l, l', e'
[state(γ, t1) |= is_at_location_from(a, l, e) & state(γ, t1) |= food_location(l)] ⇒
state(γ, t2) |= is_at_location_from(a, l', e') & state(γ, t2) |= nest_location(l')

These and a number of other properties have been formalised and using a checking software environment have been (automatically) *verified in simulation traces*. This is a first manner for verification. A second way of verification is to establish *logical relationships* between properties (by mathematical proof). This also has been performed in a number of cases. For example, under a number of assumptions the following relationships hold:

GP4 ⇒ GP5
GP3 & GP4 ⇒ GP2

The assumptions include:

- attractive routes are not branching and are not crossing each other or themselves.
- at least two ants exist for which the attractive routes end at a food location and are short enough compared to the evaporation rate of pheromones to return.
- GP5 is only valid in the infinite future, since food sources are not depleted. In practice, the simulations stop, invalidating GP5 for the ants that are still on their way to the nest.

Furthermore, an additional premise of Temporal Completion, see [12], is needed. For example, any of the following trivial (non-intended) world situations would disturb the ants: an ant comes to a location that contains a pheromone that is there without any reason (no ant dropped it), or on its way back an ant comes to a location without a pheromone (the pheromone immediately disappeared). It is clear that the above properties can only be proven under the assumption that nothing unexpected will happen. To put it differently, proofs can be given under the assumption that the set of local properties determines the whole range of events. This assumption has been added as a premise to establish the logical relationships between the properties.

6. Discussion

Clark and Chalmers [8], Section 5, provide four criteria for an extended mind: (1) the external information is a constant in the agent's life - when the information is relevant, he will rarely take action without consulting it; (2) the external information is directly

available; (3) the agent endorses retrieved external information; (4) the external information has been endorsed at some point in the past, and is there as a consequence of this endorsement. How do these criteria apply to the ants case? First, indeed an ant always senses the pheromone before choosing a direction. Second, at each location the pheromone is immediately accessible for sensing. Third, the decision for the direction is indeed always based on the pheromone. Finally, the external information is endorsed in the past: the pheromone was dropped at the direction from whence one or more ants traveled.

The extended mind perspective introduces an additional, cognitive ontology to describe properties of the physical world, which essentially is an antireductionist step, providing a more abstract and better manageable, higher level conceptualisation. For example, considering part of the external world as extended mind allows one to give another interpretation to external physical processes and states. Physical state properties such as 'pheromone is present at d' can be reconceptualised as, for example, 'the group as a whole believes that d is a relevant path'. In [15] a number of arguments can be found of why such antireductionist steps can be useful in explanation and theory development; also see Section 3 above.

In this paper, following the extended mind perspective, indeed the first steps have been made towards a high-level conceptualisation of physical processes. The main contribution of the paper is the formalisation and logical analysis of this high-level conceptualisation. The formalisation enables simulation and automated checking of dynamic properties of traces or sets of traces, and allows one to logically relate dynamic properties of different aggregation levels to each other. All this would have been more difficult in the case of an algorithmic or physically-oriented modelling perspective, involving, for example, differential equations and gradients of concentrations. As a next step, the authors are currently investigating to what extent collective processes such as ant behaviour can be interpreted and formalised as single agent processes. The first results of this research, including a formal mapping between a single agent and a multi-agent conceptualisation, are described in [5]. Moreover, work is currently in progress to model other examples of shared extended mind (outside the domain of ants). In this research, the focus is on organisms with more complex cognitive capacities (humans in particular). Meanwhile, work is in progress to elaborate the isomorphism principle mentioned in Section 3 in more detail.

Regarding details of the simulation, the authors are currently exploring whether the behaviour prescribed by attractiveness of a route can be replaced by random route selection. In addition, experiments with food sources at different distances from the nest will be undertaken to determine the relation between evaporation rate and ants finding their way home. Therefore, these food sources will be made depletive. Also, the effect of using different types of pheromones will be studied. Finally, an advanced visualisation environment is currently developed to make the simulation traces more readable.

Acknowledgements

The authors are grateful to Lourens van der Meij for his contribution to the development of the software environment, and to two anonymous referees for their comments to an earlier version of this paper.

References

[1] Barringer, H., M. Fisher, D. Gabbay, R. Owens, & M. Reynolds (1996). *The Imperative Future: Principles of Executable Temporal Logic*, Research Studies Press Ltd. and John Wiley & Sons.

[2] Bonabeau, J. Dorigo, M. and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1999.

[3] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. (2005). LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. In: Eymann, T., Kluegl, F., Lamersdorf, W., Klusch, M., and Huhns, M.N. (eds.), *Proceedings of the Third German Conference on Multi-Agent System Technologies, MATES'05*. Lecture Notes in AI, vol. 3550. Springer Verlag, 2005, pp. 165-178.

[4] Bosse, T., Jonker, C.M., and Treur, J. (2002). Simulation and analysis of controlled multi-representational reasoning processes. *Proc. of the Fifth International Conference on Cognitive Modelling, ICCM'03*. Universitats-Verlag Bamberg, 2003, pp. 27-32.

[5] Bosse, T., and Treur, J., Formal Interpretation and Analysis of Collective Intelligence as Individual Intelligence. In: Antunes, L., and Sichman, J.S. (eds.), *Proceedings of the Sixth International Workshop on Multi-Agent-Based Simulation, MABS'05*, 2005, pp. 51-65.

[6] Clark, A. (1997). Being There: Putting Brain, Body and World Together Again. MIT Press, 1997.

[7] Clark, A. (2001). Reasons, Robots and the Extended Mind. In: *Mind & Language*, vol. 16, 2001, pp. 121-145.

[8] Clark, A., and Chalmers, D. (1998). The Extended Mind. In: *Analysis*, vol. 58, 1998, pp. 7-19.

[9] Deneubourg, J.L., Aron S, Goss S., Pasteels J. M. and Duerinck G. (1986). Random Behavior, Amplification Processes and Number of Participants: How They Contribute to the Foraging Properties of Ants. In: *Evolution, Games and Learning: Models for Adaptation in Machines and Nature*, North Holland, Amsterdam, 1986, pp. 176-186.

[10] Dennett, D.C. (1996). Kinds of Mind: Towards an Understanding of Consciousness, New York: Basic Books.

[11] Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. Cambridge, MA: MIT Press.

[12] Engelfriet, J. Jonker, C.M., and Treur, J. (2002). Compositional verification of Multi-Agent Systems in Temporal Multi-Epistemic Logic. *Journal of Logic, Language and Information*, vol. 11, 2002, pp. 195-225.

[13] Forbus, K.D. (1984). *Qualitative process theory*. Artificial Intelligence, volume 24, number 1-3, pp. 85-168.

[14] Jonker, C.M., and Treur, J., Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

[15] Jonker, C.M., Treur, J., and Wijngaards, W.C.A., Reductionist and Antireductionist Perspectives on Dynamics. *Philosophical Psychology Journal*, vol. 15, 2002, pp. 381-409.

[16] Kim, J. (1996). *Philosophy of Mind*. Westview Press.

[17] Kirsh, D. & Maglio, P. (1994). On distinguishing epistemic from pragmatic action. *Cognitive Science*, vol. 18, 1994, pp. 513-49.

[18] Law, A.D. & Kelton, W.D. (2000). *Simulation Modeling and Analysis*. McGraw Hill.

[19] Menary, R. (ed.) (2004). *The Extended Mind*, Papers presented at the Conference *The Extended Mind - The Very Idea: Philosophical Perspectives on Situated and Embodied Cognition*, University of Hertfordshire, 2001. John Benjamins, 2004, to appear.

[20] Port, R.F., Gelder, T. van (eds.) (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, Mass.

PART V SINGLE VS. MULTIPLE AGENTS

CHAPTER 13

Formal Interpretation and Analysis of Collective Intelligence as Individual Intelligence

This chapter appeared as Bosse, T. and Treur, J. (2005). Formal Interpretation and Analysis of Collective Intelligence as Individual Intelligence. In: Antunes, L. and Sichman, J.S. (eds.), *Proceedings of the Sixth International Workshop on Multi-Agent-Based Simulation, MABS'05*, pp. 51-65.

Formal Interpretation and Analysis of Collective Intelligence as Individual Intelligence

Tibor Bosse and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, treur}@cs.vu.nl

Abstract. This paper addresses the question to what extent a process involving multiple agents that shows some form of collective intelligence can be interpreted as a single agent. The question is answered by formal analysis. It is shown for an example process how it can be conceptualised formalised and simulated in two different manners: from a single agent (or cognitive) and from a multi-agent (or social) perspective. Moreover, it is shown how an ontological mapping can be formally defined between the two formalisations, and how this mapping can be extended to a mapping of dynamic properties. Thus it is shown how collective behaviour can be interpreted in a formal manner as single agent behaviour.

1. Introduction

Many processes in the world can be conceptualised using an agent metaphor, as a single agent (or cognitive) process or a multi-agent (or social) process. Especially for processes that are distributed, it is natural to describe them as a group of interacting agents. If a group of agents acts in a coherent way, however, one is often tempted to intuitively and informally interpret the process in singular form as a collective, and, in fact, as one individual (super)agent. The question addressed in this paper is whether in certain cases such an informal interpretation of a multi-agent system, acting in a collective manner, as an individual can be supported by a formal analysis. The approach to address this question is by formally defining an interpretation mapping between a conceptualisation of a process as a multi-agent system and a conceptualisation of the same process as an individual.

The prerequisites to undertake such a formal analysis concern formalisations of the notion of agent, single agent behaviour and multi-agent behaviour, and the notion of interpretation mapping. More specifically, what is needed is a formal notion of what an agent is in the sense of

- distinctions between the agent's internal mental processes, the agent's body, and the agent's environment
- interactions and relationships between mental aspects and body aspects
- interactions and relationships between agent and environment, including interactions with other agents

Furthermore, formalisations of single agent behaviour and multi-agent behaviour are needed that cover

- the externally observable behaviour
- the underlying internal processes

Moreover, a formal notion of interpretation mapping of a single agent conceptualisation into a multi-agent conceptualisation is needed that

- maps ontological concepts describing a conceptualisation of a process from an individual perspective to ontological concepts describing a conceptualisation of the same a process from a multi-agent perspective
- covers mapping of individual mental state properties for the single agent conceptualisation to shared mental state properties for the multi-agent conceptualisation
- covers the mapping of dynamic aspects of single agent behaviour onto those of multi-agent behaviour

In this paper for these three notions formalisations are provided and used to indeed achieve a formalisation of how a collective can be formally interpreted as an individual.

The formalisation is evaluated for the case of collective behaviour of an ants colony. The intelligence shown by ant colonies are an interesting and currently often studied example of collective intelligence [1], [4], [6]. In this case by using pheromones the external world is exploited as a form of extended mind; cf. [2], [3], [5], [10], [11]. The analysis of this case study comprises on the one hand a multi-agent model, simulation based on identified local dynamic properties, and identification of dynamic properties for the overall process. On the other hand the same is done for an alternative model based on a single agent with internal mental states, and the two models are related to each other via the interpretation mapping.

In Section 2, a formalisation of basic agent concepts will be introduced. Section 3 explains, using a simple example, the idea of the basic formal ontology mapping between state properties in a single agent conceptualisation and state poperties in a multi-agent conceptualisation. In Section 4 this notion of basic interpretation mapping of state properties is applied to two conceptualisations of the more complex ant colony example, the central case study in the paper. Section 5 discusses the dynamics for the two conceptualisations of the ant colony example in more detail, which leads to formal specification of executable local dynamic properties that have been used for simulation. In Section 6 the basic interpretation mapping for state properties is extended to dynamic properties, thus obtaining an interpretation mapping between the two conceptualisations of the dynamics of the example ants colony process. Section 7 is a final discussion.

2. Basic Agent Concepts

The agent perspective entails a distinction between the following different types of ontologies:

- an ontology for *internal mental properties* of the agent A (MentOnt(A)),
- for properties of the agent's (physical) *body* (BodyOnt(A)),
- for properties of the (sensory or communication) *input* (InOnt(A))
- for properties of the (action or communication) *output* (OutOnt(A)) of the agent, and
- for properties of of the *external* world (ExtOnt(A)).

For example, the property 'the agent A feels pain' may belong to MentOnt(A), resp. BodyOnt(A), whereas 'it is raining' and 'the outside temperature is 7° C' may belong to ExtOnt(A). The agent input ontology InOnt defines state properties for received perception or communication, as an in-between step from environment or body state properties to

internal mental state properties, the agent output ontology OutOnt defines state properties that indicate initiations of actions or communications of the agent, as an in-between step from internal mental state properties to environment or body state properties. The combination of InOnt and OutOnt is the *agent interaction ontology*, defined by InteractionOnt = InOnt ∪ OutOnt.

To formalise state property descriptions of the types introduced above, ontologies are specified in a (many-sorted) first order logical format: an ontology is specified as a finite set of sorts, constants within these sorts, and relations and functions over these sorts. The example properties mentioned above then can be defined by nullary predicates (or proposition symbols) such as itsraining, or by using n-nary predicates (with n≥1) like has_pain(A) and has_temperature(environment, 7).

For a given ontology Ont, the propositional language signature consisting of all *state ground atoms* based on Ont is denoted by APROP(Ont). The *state properties* based on a certain ontology Ont are formalised by the propositions that can be made, using (using conjunction, negation, disjunction, implication) from the ground atoms. The notion of state as used here is characterised on the basis of an ontology defining a set of physical and/or mental (state) properties that do or do not hold at a certain point in time. In other words, a *state* S is an indication of which atomic state properties are true and which are false, i.e., a mapping S: APROP(Ont) → {true, false}.

To describe the internal and externally observable dynamics of the agent, explicit reference is made to time. Dynamics will be described as evolution of states over time. Dynamic properties can be formulated that relate a state at one point in time to a state at another point in time. A simple example is the following informally stated dynamic property for belief creation based on observation:

'if the agent observes at t1 that it is raining, then the agent will believe that it is raining'.

To express such dynamic properties, and other, more sophisticated ones, the sorted predicate logic *Temporal Trace Language* (TTL) is used [7]. Here, a *trace* over an ontology Ont is a time-indexed sequence of states over Ont. TTL is built on atoms referring to, e.g., traces, time and state properties. For example, 'in trace γ at time t property p holds' is formalised by state(γ, t) |= p. Here |= is a predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus. Dynamic properties are expressed by temporal statements built using the usual logical connectives and quantification (for example, over traces, time and state properties). For example, the dynamic property put forward above can be expressed in a more structured semiformal manner as:

'in any trace γ, if at any point in time t1 the agent A observes that it is raining,
then there exists a time point t2 after t1 such that at t2 in the trace the agent A believes that it is raining'.

In formalised TTL form it looks as follows:

$$\forall\gamma\ \forall t1\ [\ state(\gamma, t1) \models observes(A, itsraining) \ \Rightarrow\ \exists t2 \geq t1\ \ state(\gamma, t2) \models belief(A, itsraining)\ ]$$

Based on TTL, a simpler temporal language has been defined to specify simulation models. This language (the *leads to* language) enables to model direct temporal dependencies between two state properties in successive states. This executable format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the *leads to* language the notation $\alpha \bullet\!\!\rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [7]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

3. The Basic Interpretation Mapping

In this section it is discussed how a conceptualisation based on a single agent and individual (internal) mental state properties can formally be mapped onto a conceptualisation based on multiple agents and shared (for the sake of simplicity assumed external) mental state properties. Here this ontological mapping is only given in its basic form, for the state properties. In Section 6 the basic mapping is extended to temporal expressions describing behaviour.

First, consider Figure 1. This figure depicts a simple case of a single agent A with behaviour based on an individual internal mental state property m1. The solid arrows depict temporal *leads to* relationships. Mental state property m1 (temporally) depends on observations of three world state properties c1, c2, c3. Moreover, action a1 depends on m1.

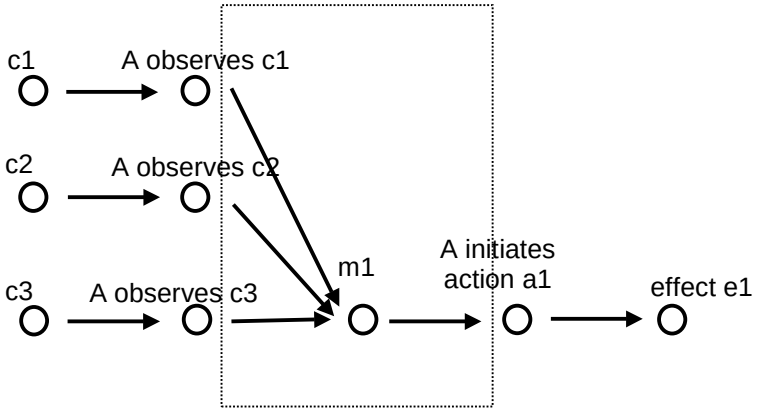


Figure 1. Single Agent behaviour based on an internal mental state

Now consider Figure 2. This figure depicts a group of agents A1, A2, A3, A4 with behaviour based on a physical external world state property m2 that serves as a shared external mental state property. To create this shared mental state property, actions a2a, a2b, a2c of the agents A1, A2, A3 are needed, and to show the behaviour, first an observation of m2 by agent A4 is needed. Note that here the internal processing is chosen as simple as possible: stimulus response. The interaction between agent and external world is a bit more complex: compared to a single agent perspective with internal mental state m1, extra actions of some of the agents needed to create the external mental state property m2, and additional observations are needed to observe it.

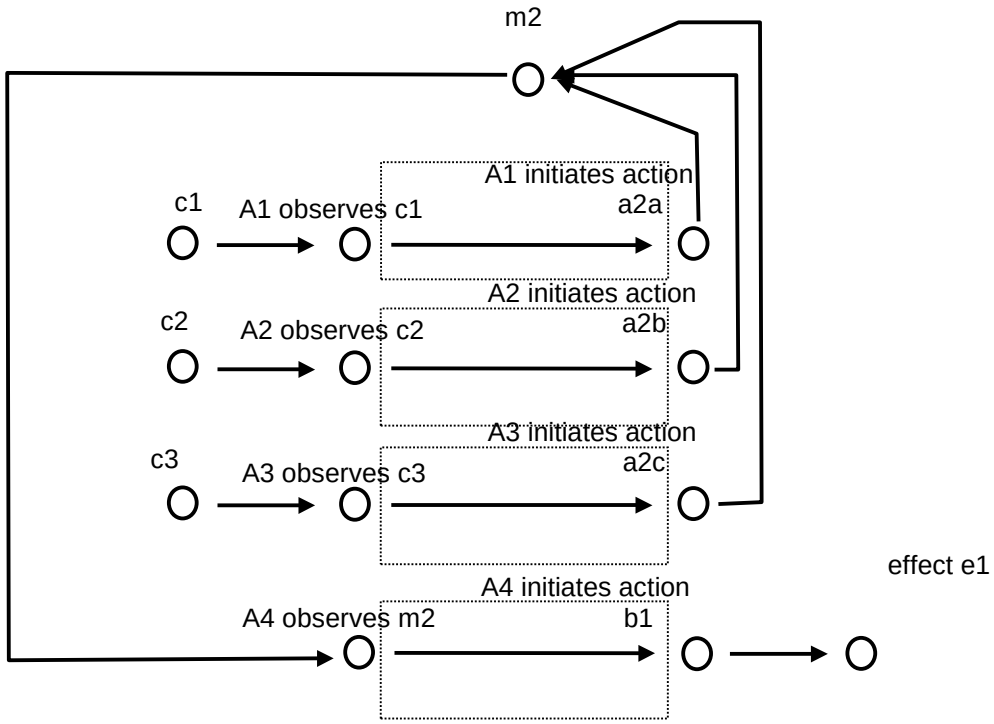


Figure 2. Multi-Agent behaviour based on a shared external mental state

To make the similarity between the two different cognitive processes more precise, the following mapping from the nodes (state properties) in Figure 1 onto nodes in Figure 2 can be made (see Figure 3):

External world state properties

φ: c1	→	c1
φ: c2	→	c2
φ: c3	→	c3
φ: effect e1	→	effect e1

Observation state properties

φ: A observes c1	→	A1 observes c1
φ: A observes c2	→	A2 observes c2
φ: A observes c3	→	A3 observes c3

Action initiation state properties

φ: A initiates action a1	→	A4 initiates action b1

Mental state property to external world state property

φ: m1	→	m2

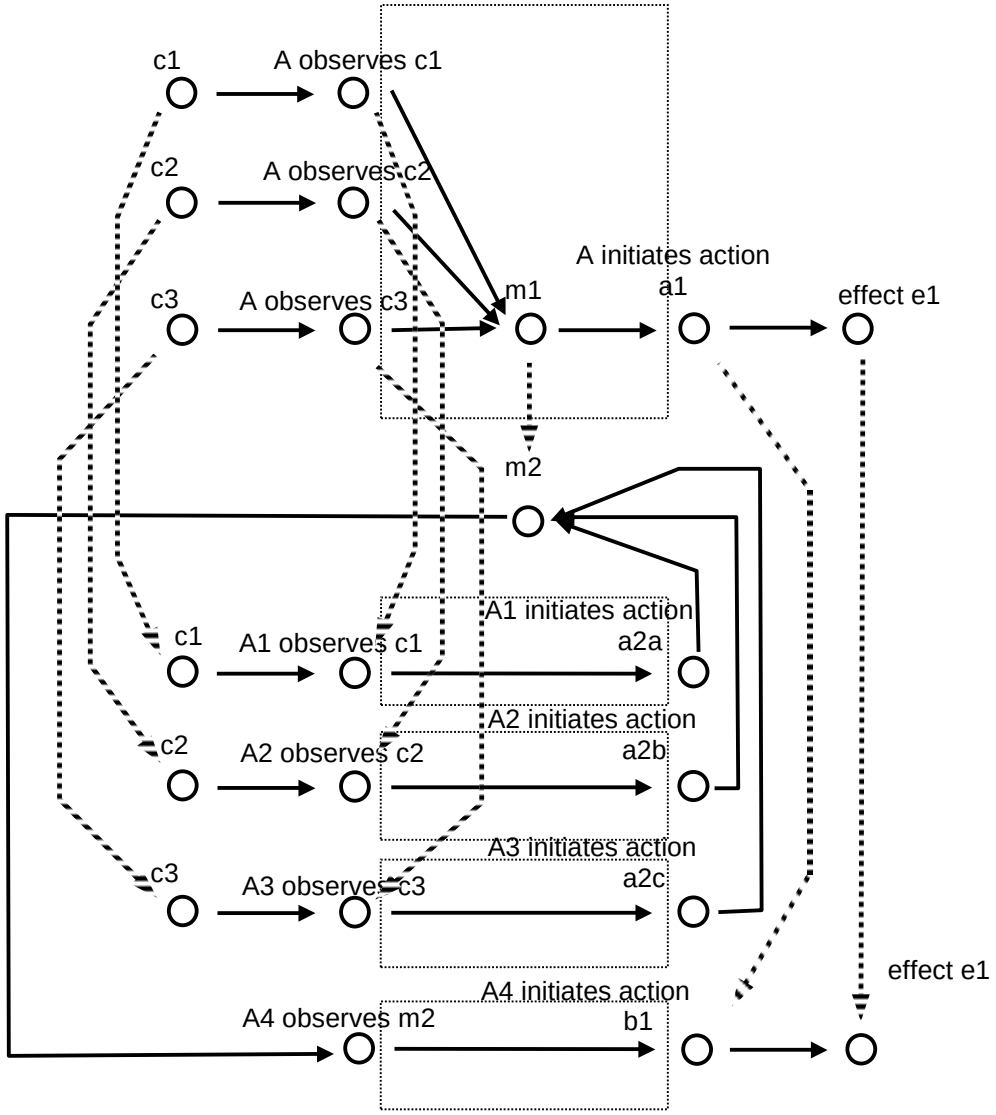


Figure 3. Isomorphism relationship between shared extended mind and individual mental state

Note that in this case, for simplicity it is assumed that each observation of A is an observation of exactly one of the Ai, and the same for actions.

This mapping φ, indicated by the vertical dotted arrows in Figure 3, preserves the temporal dependencies in the form of *leads to* relationships (the solid arrows) and provides an (isomorphic, in the mathematical sense) embedding of a cognitive process based on internal mind into a cognitive process based on extended mind.

In their paper about extended mind, Clark and Chalmers [3] point at the similarity between cognitive processes in the head and some processes involving the external world. This similarity can be used as an indication that these processes can be considered extended cognitive processes or extended mind:

> *If, as we confront some task, a part of the world functions as a process which,* were it done in the head*, we would have no hesitation in recognizing as part of the cognitive process, then that part of the world* is *(so we claim) part of the cognitive process. Cognitive processes ain't (all) in the head! [3], Section 2. (…)*

One can explain my choice of words in Scrabble, for example, as the outcome of an extended cognitive process involving the rearrangement of tiles on my tray. Of course, one could always try to explain my action in terms of internal processes and a long series of "inputs" and "actions", but this explanation would be needlessly complex. If an isomorphic process were going on in the head, we would feel no urge to characterize it in this cumbersome way. (…) In a very real sense, the re-arrangement of tiles on the tray is not part of action; it is part of thought*. [3], Section 3.*

Clark and Chalmers [3] use the isomorphism to a process 'in the head' as one of the criteria to consider external and interaction processes as cognitive, or mind processes. As the shared mental state property m2 is modelled as an external state property, this 'isomorphism principle' is formalised in Figure 3 for a simple example of such an isomorphism. Note that the process from m1 to action a1, modelled as one step in the single agent, internal case, is mapped onto a process from m2 via A4 observes m2 to A4 initiates action b1, in the external case modelled as a two-step process. So the isomorphism is an embedding in one direction, not a bidirectional isomorphism, simply because on the multi-agent side, the observation state for A4 observing m2 has no counterpart in the single agent, internal case (and the same for the agents A1, A2, A3 initiating actions a2a, a2b, a2c).

Notice that the mapping φ is a (formal) mapping between state properties. However, it was already put forward that temporal *leads to* relations are preserved under φ, so the mapping can be extended to a mapping of *leads to* properties onto *leads to* properties. From a more general perspective, it can be analysed how far the mapping φ can be extended to a (formal) mapping from dynamic properties to dynamic properties expressed in TTL. This will be addressed in detail in Section 6.

4. Two Conceptualisations and their Mapping

The general formalisation perspective put forward in previous sections has been evaluated for a case study: a process of collective ant behaviour. For this example process two conceptualisations have been made, one from a multi-agent (or social) perspective, and one for a single agent (or cognitive) perspective.

The world in which the ants live is described by a labeled graph as depicted in Figure 4. Locations are indicated by A, B,…, and edges by E1, E2,… To represent such a graph the predicate connected_to_via(l0,l1,e1) is used. The ants move from location to location via edges; while passing an edge, pheromones are dropped. The same or other ants sense these pheromones and follow the route in the direction of the strongest concentration. Pheromones evaporate over time; therefore such routes can vary over time. The goal of the ants is to find food and bring this back to their nest. In this example there is only one nest (location A) and one food source (location F).

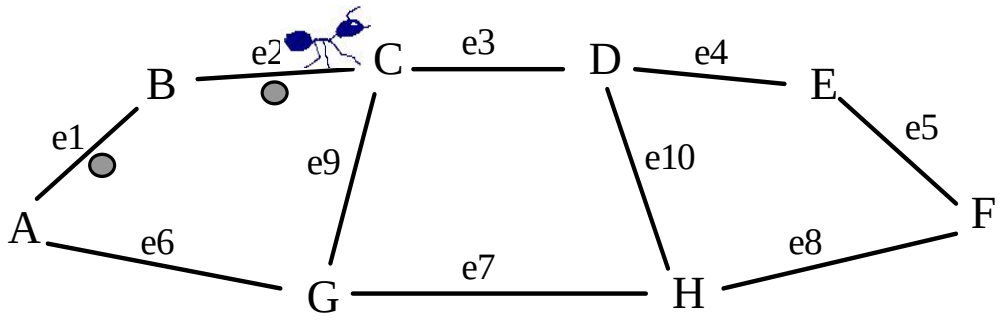


Figure 4. An ants world

4.1. Multi-Agent Conceptualisation

The example process conceptualised from a multi-agent perspective concerns multiple agents (the ants), each of which has input (to observe) and output (for moving and dropping pheromones) states, and a physical body which is at certain positions over time, but no internal mental state properties (they are assumed to act purely by stimulus-response behaviour). An overview of the formalisation of the state properties of this multi-agent conceptualisation is shown in Table 1.

	Multi-Agent Conceptualisation
	body positions in world:
pheromone level at edge e is i	pheromones_at(e, i)
ant a is at location l coming from e	is_at_location_from(a, l, e)
ant a is at edge e to l2 coming from location l1	is_at_edge_from_to(a, e, l1, l2)
ant a is carrying food	is_carrying_food(a)
	world state properties:
edge e connects location l1 and l2	connected_to_via(l1, l2, e)
location l has i neighbours	neighbours(l, i)
edge e is most attractive for ant a coming from location l	attractive_direction_at(a, l, e)
	input state properties:
ant a observes that it is at location l coming from edge e	observes(a, is_at_location_from(l, e))
ant a observes that it is at edge e to l2 coming from location l1	observes(a, is_at_edge_from_to(e, l1, l2))
ant a observes that edge e has pheromone level i	observes(a, pheromones_at(e, i))
	output state properties:
ant a initiates action to go to edge e to l2 coming from location l1	to_be_performed(a, go_to_edge_from_to(e, l1, l2))
ant a initiates action to go to location l coming from edge e	to_be_performed(a, go_to_location_from(l, e))
ant a initiates action to drop pheromones at edge e coming from location l	to_be_performed(a, drop_pheromones_at_edge_from(e, l))
ant a initiates action to pick up food	to_be_performed(a, pick_up_food)
ant a initiates action to drop food	to_be_performed(a, drop_food)

Table 1. Multi-Agent conceptualisation: state properties

4.2. Single-Agent Conceptualisation

The conceptualisation of the example process from a single agent perspective (Superant S), however, takes into account one body, of which each ant is part (for convenience we call them the 'paws' of this body). Also the pheromone levels at the edges are part of the body.

The body position of this agent in the world is defined by the collection of positions of each of the paws. Mental state properties for this single agent occur in the form of beliefs that a certain edge has a certain relevance level (realised in the body by the pheromone levels). Input of the single agent is defined by the collection of inputs of the ants at each of the paws. Output is defined by initiation of movements of one or more of the paws. Notice that in this case dropping pheromones is not an action, but an internal body process to create or update the proper beliefs by creating or updating their realisation in the body.

An overview of the formalisation of the state properties of the multi-agent conceptualisation is shown in Table 2. Note that there S stands for the Superant.

Single Agent Conceptualisation	
mental state properties:	
belief(S, relevance_level(e, i))	belief on the relevance level i of an edge e
body position in world:	
has_paw_at_location_from(S, p, l, e)	position of paw p at location l coming from edge e
has_paw_at_edge_from_to(S, p, e, l1, l2)	position of paw p at edge e to l2 coming from location l1
is_carrying_food_with_paw(S, p)	paw p is carrying food
world state properties:	
connected_to_via(l1, l2, e)	edge e connects location l1 and l2
neighbours(l, i)	location l has i neighbours
attractive_direction_at(p, l, e)	edge e is most attractive for paw p coming from location l
input state properties:	
observes(S, has_paw_at_location_from(p, l, e))	S observes that paw p is at location l coming from edge e
observes(S, has_paw_at_edge_from_to(p, e, l1, l2))	S observes that paw p is at edge e to l2 coming from location l1
output state properties:	
to_be_performed(S, move_paw_to_edge_from_to(p, e, l1, l2))	S initiates action to move paw p from location l1 to edge e to l2
to_be_performed(S, move_paw_to_location_from (p, l, e))	S initiates action to move paw p from edge e to location l
to_be_performed(S, pick_up_food_with_paw(p))	S initiates action to pick up food with paw p
to_be_performed(S, drop_food_with_paw(p))	S initiates action to drop food with paw p

Table 2. Single Agent conceptualisation: state properties

4.3. Mapping between Conceptualisations

The two conceptualisations described in Sections 4.1 and 4.2 are two conceptualisations of one and the same example process. A concept in any of the two conceptualisations in principle has a one-to-one correspondence to an aspect of this example process which can be considered the informal semantics of the concept (in our case the concept is formalised); see the double arrows in Figure 5.

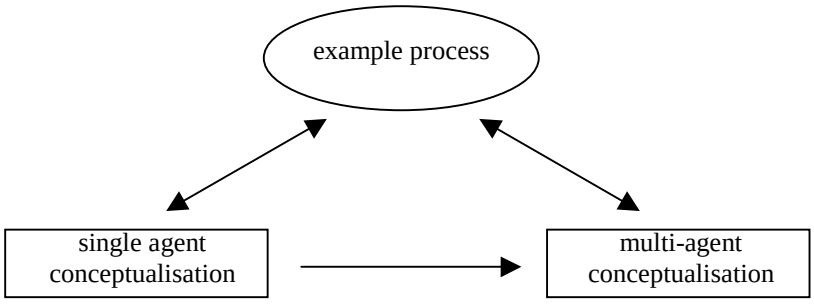


Figure 5. Two conceptualisations and their mapping

Given these one-to-one correspondences, a mapping from the single agent conceptualisation to the multi-agent conceptualisation can be made as follows:

1) Take any state property c belonging to the single agent conceptualisation
2) Identify to what aspect a of the example process this state property corresponds
3) Identify to which state property d in the multi-agent conceptualisation this aspect a corresponds
4) Map c to d.

If this approach works, then a mapping is obtained that is sincere with respect to the example process: the state property d to which c is mapped corresponds to the same aspect a of the process as c, and therefore will be true (for the informal semantics) if and only if c is. The approach can also fail. It can fail in 2) if state properties are used in the single agent conceptualisation that have no counterpart in the example process. It can fail in 3) if in the single agent conceptualisation aspects of the process are covered that are left out of consideration in the other conceptualisation. Actually such aspects exist the other way around: there are aspects of the process, such as observing the pheromones covered by the multi-agent conceptualisation, but not by the single agent conceptualisation. Therefore such a mapping is not possible from right to left in Figure 5 (see also Figure 3 in Section 3, where the mapping is not bijective either). However, a mapping from left to right (single agent to multi-agent conceptualisation), is possible. It is shown in Table 3. Note that there S stands for the Superant, and paw p corresponds to ant a.

Single Agent Conceptualisation	**Multi-Agent Conceptualisation**
belief(S, relevance_level(e, i))	pheromones_at(e, i)
has_paw_at_location_from(S, p, l, e) has_paw_at_edge_from_to(S, p, e, l1, l2) is_carrying_food_with_paw(S, p)	is_at_location_from(a, l, e) is_at_edge_from_to(a, e, l1, l2) is_carrying_food(a)
connected_to_via(l1, l2, e) neighbours(l, l) attractive_direction_at(p, l, e)	connected_to_via(l1, l2, e) neighbours(l, i) attractive_direction_at(a, l, e)
observes(S, has_paw_at_location_from(p, l, e)) observes(S, has_paw_at_edge_from_to(p, e, l1, l2)) ---	observes(a, is_at_location_from(l, e)) observes(a, is_at_edge_from_to(e, l1, l2)) observes(a, pheromones_at(e, i))
to_be_performed(S, move_paw_to_edge_from_to(p, e, l1, l2)) to_be_performed(S, move_paw_to_location_from (p, l, e)) --- to_be_performed(S, pick_up_food_with_paw(p)) to_be_performed(S, drop_food_with_paw(p))	to_be_performed(a, go_to_edge_from_to(e, l1, l2)) to_be_performed(a, go_to_location_from(l, e)) to_be_performed(a, drop_pheromones_at_edge_from(e, l)) to_be_performed(a, pick_up_food) to_be_performed(a, drop_food)

Table 3. Mapping between state properties

5. Two Simulation Models

The two conceptualisations introduced above have been used to create two simulation models for collective ant behaviour: one from a multi-agent (social) perspective and one from a single agent (cognitive) perspective. The basic building blocks of the model were dynamic properties in *leads to* format, specifying the local mechanisms of the process. Examples of such local dynamic properties (for the *multi-agent case*) are the following:

LP5 (Selection of Edge)
"If an ant observes that it is at location l, and there are three edges connected to that location, then the ant goes to the edge with the highest amount of pheromones."
observes(a, is_at_location_from(l, e0)) and neighbours(l, 3) and connected_to_via(l, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(l, l2, e2) and observes(a, pheromones_at(e2, i2)) and e0 ≠ e1 and e0 ≠ e2 and e1 ≠ e2 and i1 > i2 •→→ to_be_performed(a, go_to_edge_from_to(e1, l1))

LP6 (Arrival at Edge)
"If an ant goes to edge e from location l to location l1, then later the ant will be at this edge e."
to_be_performed(a, go_to_edge_from_to(e, l, l1)) •→→ is_at_edge_from_to(a, e, l, l1)

LP9 (Dropping of Pheromones)
"If an ant observes that it is at an edge e from a location l to a location l1, then it will drop pheromones at this edge e."
observes(a, is_at_edge_from_to(e, l, l1)) •→→ to_be_performed(a, drop_pheromones_at_edge_from(e, l))

LP12 (Observation of Pheromones)
"If an ant is at a certain location l, then it will observe the number of pheromones present at all edges that are connected to location l."
is_at_location_from(a, l, e0) and connected_to_via(l, l1, e1) and pheromones_at(e1, i) •→→ observes(a, pheromones_at(e1, i))

LP13 (Increment of Pheromones)
"If an ant drops pheromones at edge e, and no other ants drop pheromones at this edge, then the new number of pheromones at e becomes i*decay+incr." Here, i is the old number of pheromones, decay is the decay factor, and incr is the amount of pheromones dropped.
to_be_performed(a1, drop_pheromones_at_edge_from(e, l1)) and ∀l2 not to_be_performed(a2, drop_pheromones_at_edge_from(e, l2)) and ∀l3 not to_be_performed(a3, drop_pheromones_at_edge_from(e, l3)) and a1 ≠ a2 and a1 ≠ a3 and a2 ≠ a3 and pheromones_at(e, i) •→→ pheromones_at(e, i*decay+incr)

LP14 (Collecting of Food)
"If an ant observes that it is at location F (the food source), then it will pick up some food."
observes(a, is_at_location_from(F, e)) •→→ to_be_performed(a, pick_up_food)

To model the example from a single agent perspective, again a number of local dynamic properties are used. Most, but not all of these local properties have a 1:1 correspondence to those for the multi-agent case. For example, the properties for the *single agent case* that correspond to the properties above are as follows (see the next section for more information about this correspondence):

LP5' (Selection of Edge)
"If S observes that it has a paw p at location A, and there are three edges connected to that location, then S will move its paw to the edge of which it believes that it has the highest relevance level."
observes(S, has_paw_at_location_from(p, l, e0)) and neighbours(l, 3) and connected_to_via(l, l1, e1) and belief(S, relevance_level(e1, i1)) and connected_to_via(l, l2, e2) and belief(S, relevance_level(e2, i2)) and e0 ≠ e1 and e0 ≠ e2 and e1 ≠ e2 and i1 > i2 •→→ to_be_performed(S, move_paw_to_edge_from_to(p, e1, l1))

LP6' (Paw Arrival at Edge)
"If S moves its paw p to an edge e from a location l to a location l1, then later this paw will be at this edge e."
to_be_performed(S, move_paw_to_edge_from_to(p, e, l, l1)) •→→ has_paw_at_edge_from_to(S, p, e, l, l1)

LP11' (Increment of Belief)
"If S has exactly one paw at edge e, then the new number of pheromones at e becomes i*decay+incr."
observes(S, has_paw_at_edge_from_to(p1, e, l, l1)) and
∀l2 not observes(S, has_paw_at_edge_from_to(p2, e, l, l2)) and
∀l3 not observes(S, has_paw_at_edge_from_to(p3, e, l, l3)) and
p1 ≠ p2 and p1 ≠ p3 and p2 ≠ p3 and belief(S, relevance_level(e, i)) •→→
belief(S, relevance_level(e, i*decay+incr))

LP12' (Collecting of Food)
"If S observes that it has a paw p at location F (the food source), then it will pick up some food with that paw."
observes(S, has_paw_at_location_from(p, F, e)) •→→
to_be_performed(S, pick_up_food_with_paw(p))

A special software environment has been created to enable the simulation of executable models. Based on an input consisting of dynamic properties in *leads to* format, it can generate simulation traces. An example of (part of) such a trace can be seen in Figure 6. Time is on the horizontal axis, the state properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false. This trace was based on the multi-agent simulation model.

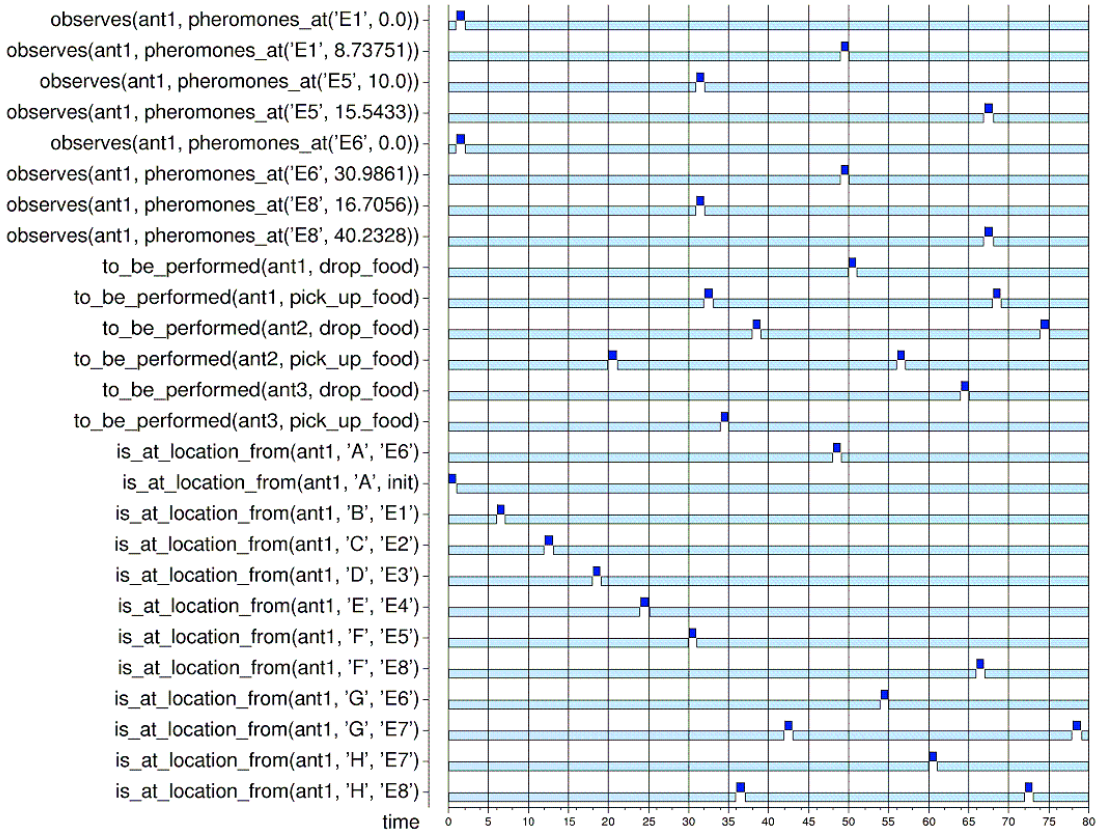


Figure 6. Multi-Agent Simulation Trace

Figure 7 depicts a similar trace as Figure 6, this time based on the single agent simulation model. Note that there are several differences between Figure 6 and 7. In the first place, all ants that are treated as separate agents in Figure 6, are considered as parts of Superant S in Figure 7. For example, is_at_location_from(ant1, A, E6)) in the multi-agent case corresponds to has_paw_at_location_from(S, paw1, A, E6)) in the single agent case.

Another important difference is that in the single agent case, there is no explicit observation of pheromones. The reason for this is that the belief(S, relevance_level(e, i)) states (which are the single agent equivalent for the pheromones_at(e, i) states in the multi-agent case) are internal states of S, which do not have to be observed.

Altogether, the software environment has been used to successfully generate a large number of simulation traces on the basis of both simulation models. To limit complexity, only some fragments of such traces are shown here. However, all complete traces, as well as the complete sets of dynamic properties, are shown on the following URL: http://www.cs.vu.nl/~tbosse/isomorphism/.

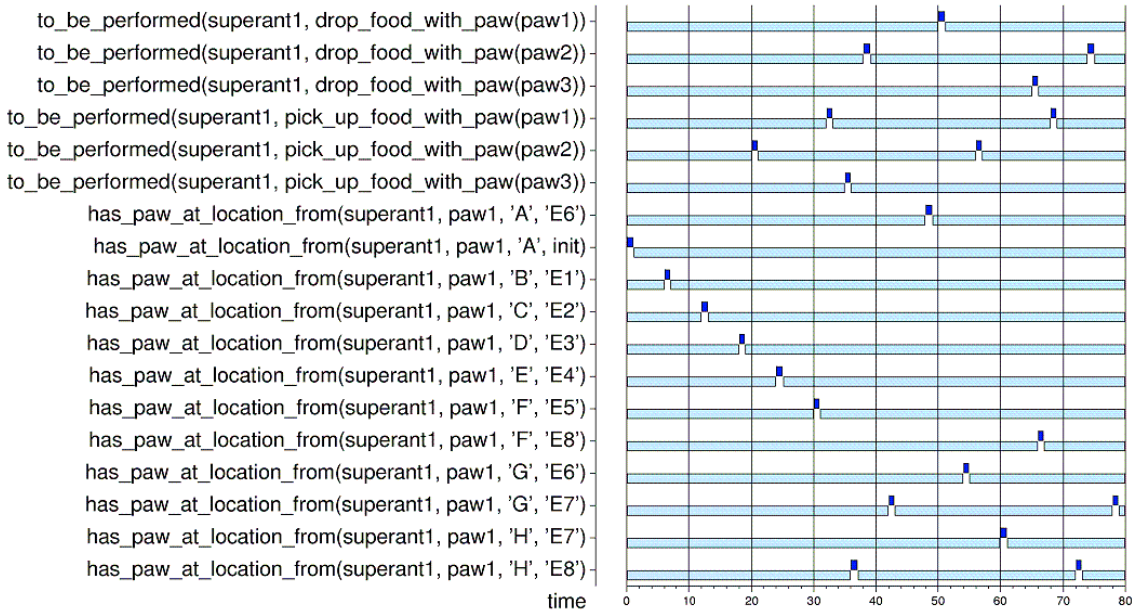


Figure 7. Single Agent Simulation Trace

6. The Extended Interpretation Mapping

In Section 3 it was shown how the basic interpretation mapping can be defined as a mapping between state properties. It was suggested that this mapping can be extended to a mapping between local dynamic properties in *leads to* format. Therefore, the following interpretation mapping can be defined:

$$\varphi(\alpha \bullet\!\!\rightarrow\!\!\!\!\rightarrow \beta) = \varphi(\alpha) \bullet\!\!\rightarrow\!\!\!\!\rightarrow \varphi(\beta)$$

Using this interpretation mapping, combined with the basic mapping of the state ontology elements described in Section 4, mappings between the dynamic properties of the case study can be found, e.g.:

φ(LP6')
= φ(to_be_performed(S, move_paw_to_edge_from_to(p, e, l, l1)) •→→
has_paw_at_edge_from_to(S, p, e, l, l1))
= φ(to_be_performed(S, move_paw_to_edge_from_to(p, e, l, l1))) •→→
φ(has_paw_at_edge_from_to(S, p, e, l, l1))
= to_be_performed(a, go_to_edge_from_to(e, l, l1)) •→→ is_ at_edge_from_to(a, e, l, l1))
= LP6

A mapping between all local dynamic properties (in *leads to* format) of the case study is given in Table 4. Notice that in some cases a certain dynamic property is mapped to a dynamic property that is not literally in the multi-agent model, but actually is a combination of two other local properties present in the model. This shows where the single agent conceptualisation is simpler than the multi-agent conceptualisation.

Single Agent Conceptualisation	Multi-Agent Conceptualisation
LP1'	LP1
LP2'	LP2
LP3'	LP3
LP4'	LP4
LP5'	LP5 & LP12
LP6'	LP6
LP7'	LP7
LP8'	LP8
LP9'	LP10
LP10'	LP11
LP11'	LP9 & LP13
LP12'	LP14
LP13'	LP15
LP14'	LP16
LP15'	LP17
LP16'	LP9 & LP18

Table 4. Mapping between local dynamic properties

In addition, it is possible to extend the mapping to the wider class of TTL expressions. Recall that TTL expressions are built on atoms of the form state(γ, t) |= p. By the basic mapping the state property p can be translated into φ(p), which is assumed to be part of the ontology of one of the agents Ai in the multi-agent conceptualisation. Moreover, the trace name γ can be mapped onto a trace name $\varphi(\gamma) = \gamma'$. Then the extended interpretation mapping for state(γ, t) |= p is defined by:

$$\varphi\text{: state}(\gamma, t) \models p = \text{state}(\gamma', t) \models \varphi(p)$$

After these atoms have been mapped, TTL expressions as a whole can be mapped in a straightforward compositional manner:

$$\varphi(A\&B) = \varphi(A) \,\&\, \varphi(B)$$
$$\varphi(A \Rightarrow B) = \varphi(A) \Rightarrow \varphi(B)$$
$$\varphi(\text{not } A) = \text{not } \varphi(A)$$
$$\varphi(\forall v\, A(v)) = \forall v'\, \varphi(A(v'))$$
$$\varphi(\exists v\, A(v)) = \exists v'\, \varphi(A(v'))$$

For example, take the following TTL expression, which is a global property for the single agent case of the ant example:

GP1' Food Discovery
"Eventually, one of the paws of S will be at the food location."
$\exists t,p,l,e$ [state(γ, t) |= has_paw_at_location_from(S, p, l, e) & state(γ, t) |= food_location(l)]

This expression is mapped as follows:

$\varphi(\exists t,p,l,e$ [state(γ, t) |= has_paw_at_location_from(S, p, l, e) & state(γ, t) |= food_location(l)])
= $\exists t',p',l',e'$ φ([state(γ, t') |= has_paw_at_location_from(S, p', l', e') &
state(γ, t') |= food_location(l')])
= $\exists t',p',l',e'$ [φ(state(γ, t') |= has_paw_at_location_from(S, p', l', e')) &
φ(state(γ, t') |= food_location(l'))]
= $\exists t',p',l',e'$ [state(γ', t') |= φ(has_paw_at_location_from(S, p', l', e')) &
state(γ', t') |= φ(food_location(l'))]
= $\exists t',p',l',e'$ [state(γ', t') |= is_at_location_from(p', l', e') & state(γ', t') |= food_location(l')]

7. Discussion

This paper addresses the question to what extent a process involving multiple agents that shows some form of collective intelligence can be interpreted as single agent behaviour. The question is answered by formal analysis. It is shown for an example process how it can be conceptualised and formalised in two different manners: from a single agent (or cognitive) and from a multi-agent (or social) perspective. Moreover, it is shown how a basic ontological mapping can be formally defined between the two formalisations, and how this mapping can be extended to a mapping of dynamic properties. Thus it is shown how the collective behaviour can be interpreted in a formal manner as single agent behaviour. For example, the fact that food is taken from the source to the nest can be explained by a sequence of actions of one agent, based on its beliefs.

Having such a mapping allows one to explain collective or social behaviour in terms of single agent concepts in the following manner. Behaviour often is explained by considering the basic underlying causal relations or mechanisms. The mapping and its formalisation allows to replace an explanation of behaviour in terms of basic mechanisms involving frequent interactions of the multiple agents (with each other and/or with the external world), by an explanation that leaves out these interactions and bases itself directly on mental states of the single agent conceptualisation. This explanation is simpler, more abstract and perhaps more elegant, than the more complicated explanation based on the interactions. This is made possible by introducing a new ontology for states involved. For example, considering part of the external world as extended mind allows one to give another interpretation to external physical processes and states. Physical state properties such as 'pheromone is present at d' are reconceptualised as, for example, 'it is believed that d is a relevant path'. Why would one introduce extra language to refer to the same fact in the world? Given the literature on reduction, where often it is claimed that mental state properties can be and actually should be replaced by their physical realisers, at first sight such an opposite move may seem a bit surprising. For example, Kim [9] (pp. 214-216) claims that ontological simplification is one of the reasons to reduce mental state properties to physical state properties. In the extended mind case at hand the converse takes place; a question is what is the advantage of this ontological complication. A number of arguments in support of this can be given. By Clark and Chalmers [3], it is claimed that this allows application of other types of explanation and other methods of scientific investigation:

> *(…) we allow a more natural explanation of all sorts of actions. (…) in seeing cognition as extended one is not merely making a terminological decision; it makes a significant difference to the methodology of scientific investigation. In effect, explanatory methods that might once have been*

thought appropriate only for the analysis of "inner" processes are now being adapted for the study of the outer, and there is promise that our understanding of cognition will become richer for it. [3], Section 3.

In [8] it is explained in some detail why in various cases in other areas (such as Computer Science) such an antireductionist strategy often pays off; some of the discussed advantages in terms of insight, transparency and genericity are: additional higher-level ontologies can improve understanding as they may allow simplification of the picture by abstracting from lower-level details; more insight is gained from a conceptually higher-level perspective; analysis of more complex processes is possible; finally, the same concepts have a wider scope of application, thus obtaining unification.

Future research will further analyse the interpretation mapping in the context of logic: the notion of an interpretation of one (formal) logical theory T in another logical theory T' has a formal definition in logic. It is an interesting question whether it can be proven logically that the conditions of this definition are fulfilled for the mapping defined in this paper. For example, a question is whether it can be proven that:

$$T \vdash \alpha \Rightarrow T' \vdash \varphi(\alpha)$$

for all formulae α, where T is a logical theory of single agent behaviour and T' a theory of multi-agent behaviour.

Acknowledgements

The authors are grateful to Catholijn Jonker for many valuable discussions on the topic of the paper, to Lourens van der Mey for his contribution to the development of the software environment, and to Martijn Schut for his contribution to parts of the simulations.

References

[1] Bonabeau, E., Dorigo, M., and Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York.

[2] Clark, A. Being There: Putting Brain, Body and World Together Again. MIT Press, 1997.

[3] Clark, A. and Chalmers, D. The Extended Mind. In: *Analysis*, vol. 58, 1998, pp. 7-19.

[4] Deneubourg, J.L., Aron, S., Goss, S., Pasteels, J.M., and Duerinck, G. Random Behavior, Amplification Processes and Number of Participants: How They Contribute to the Foraging Properties of Ants. In: *Evolution, Games and Learning: Models for Adaptation in Machines and Nature*, North Holland, Amsterdam, 1986, pp. 176-186.

[5] Dennett, D.C. Kinds of Mind: Towards an Understanding of Consciousness, New York: Basic Books.

[6] Drogoul, A., Corbara, B., and Fresneau, D.: MANTA: New experimental results on the emergence of (artificial) ant societies. In: Gilbert N. and Conte R. (eds.), *Artificial Societies: the computer simulation of social life*, UCL Press, 1995.

[7] Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. *Cognitive Systems Research Journal*, vol. 4(3), 2003, pp. 191-210.

[8] Jonker, C.M., Treur, J., and Wijngaards, W.C.A., Reductionist and Antireductionist Perspectives on Dynamics. *Philosophical Psychology Journal*, vol. 15, 2002, pp. 381-409.

[9] Kim, J. *Philosophy of Mind*. Westview Press

[10] Kirsh, D. and Maglio, P. On distinguishing epistemic from pragmatic action. *Cognitive Science*, vol. 18, 1994, pp. 513-49.

[11] Menary, R. (ed.) (2005). *The Extended Mind*, Papers presented at the Conference *The Extended Mind - The Very Idea: Philosophical Perspectives on Situated and Embodied Cognition*, University of Hertfordshire, 2001. John Benjamins, 2005, to appear.

PART V SINGLE VS. MULTIPLE AGENTS

CHAPTER 14

Organisation Modelling for the Dynamics of Complex Biological Processes

This chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2004). Organisation Modelling for the Dynamics of Complex Biological Processes. In: Lindemann, G., Moldt, D., and Paolucci, M. (eds.), *Proceedings of the International Workshop on Regulated Agent-Based Social Systems: Theories and Applications, RASTA'02*, Lecture Notes in Artificial Intelligence, vol. 2934, Springer Verlag, pp. 92-112.

Organisation Modelling for the Dynamics of Complex Biological Processes

Tibor Bosse, Catholijn M. Jonker, and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email : {tbosse, jonker, treur}@cs.vu.nl
URL : http://www.cs.vu.nl/~{tbosse, jonker, treur}

Abstract. This paper shows how an organisation modelling approach can be used to model the dynamics of biological organisation, in particular the circulatory system in biological organisms (mammals). This system consists of a number of components that are connected and grouped together. Dynamic properties at different levels of aggregation of this organisation model have been identified, and interlevel relationships between these dynamic properties at different aggregation levels were made explicit. Based on the executable properties simulation has been performed and properties have been checked for the produced simulation traces. Thus the logical relationships between properties at different aggregation levels have been verified. Moreover, relationships between roles within the organisation model and realisers of these roles have been defined. This case study shows that within biological and medical domains organisation modelling techniques can play a useful role in modelling complex systems at a high level of abstraction.

1. Introduction

In biological systems often many complex distributed interacting processes take place, that together result in some form of coherent joint action. Examples of such biological systems are mammals, insect colonies and bacteria. During evolution, Nature has developed several forms of organisational structure; typical examples are the organisation of a beehive, the coordinated processes of organs in mammals, and the well-organised regulated biochemistry of a living cell. Usually such biological systems are addressed by modelling the underlying physical/chemical processes by mathematical and system theoretical techniques, for example sets of differential equations; e.g., [26]. For some small unicellular organisms, a few isolated chemical pathways are understood in sufficient kinetic detail to obtain a description (by differential equations) of their import and primary processing of nutrients; e.g., in *Escherichia coli* [22], [24], or yeast [21]. However, even if all details would be available, at best this approach provides a description that is inherently low-level and complex. The adequacy of such mathematical techniques addressing the underlying physical/chemical level can be questioned. Such approaches do not exploit the apparent organisational structure that can be identified at a conceptual level within the biological systems addressed; the types of techniques often used are not tuned to modelling at such a conceptual level of the organisation of the distributed interacting processes.

In the area of organisation modelling, to handle complex distributed dynamics of the interaction between multiple agents in human society, often some type of organisational structure is exploited. The dynamics that emerge from multiple interacting agents within human society has been studied within Social Sciences in the area of Organisation Theory (e.g., [12], [13], [17], [19]) and within Artificial Intelligence in the area of Agent Systems

(e.g., [2], [25]). To manage complex, decentralised dynamics in human society, organisational structure is a crucial element: organisation provides a structuring and co-ordination of the processes in such a manner that a process or agent involved can function in a more adequate manner. The dynamics shown by a given organisational structure is much more dependable than in an entirely unstructured situation. To exploit such organisational structures in a society particularly in modelling of these processes, within the agent systems area a number of conceptual modelling approaches have been developed, where a specific form of organisational structure is taken as a central concept. One of the recently developed organisational modelling approaches is the Agent/Group/Role (AGR) approach introduced in [3], extended with operational semantics in [4], and with a specification language for dynamic properties in [5].

Like in human societies, as discussed above, many biological systems take the form of complex organised distributed interacting processes. Therefore a natural research question addressed in this paper is whether organisational modelling techniques provide adequate means to model such biological systems at a conceptual-organisational level. If such an approach succeeds, it may be expected that it results in models of a much higher level than those addressing the biological processes at the level of their physiology or chemistry. A relating hypothesis is that such higher-level models can be simulated and analysed much more easily than the more complex mathematical models. These are the issues addressed in this paper. To explore these issues, in a rather arbitrary manner one specific available organisation modelling framework has been chosen and one specific organised biological phenomenon on which this organisation modelling framework was applied.

The chosen organisation modelling framework is the one described in [10], addressing both analysis and simulation of AGR-models, and supported by a software environment; a formal foundation can be found in [10]. This dynamic modelling environment allows to

- specify dynamic properties for the different elements and levels of aggregation within an AGR organisation model
- relate these dynamic properties to each other according to the organisational structure
- use dynamic properties in executable form as a declarative specification of a simulation model and perform simulation experiments
- automatically check dynamic properties for simulated or empirical traces

The goal of this paper is, in particular, to illustrate how this dynamic modelling framework for organisations, whilst being a conceptual approach, can also be used to model complex organised dynamics in biological systems involving several interacting processes.

The chosen case study for such a biological system, concerns the most primary dynamics of the circulatory system in biological organisms (mammals in particular). This biological system shows sufficient complexity to be an interesting challenge. In the literature, many different kinds of cardiovascular (CV) models exist, typically based on modelling the physiology by differential equations. The first modern CV models were based on the *Windkessel* theory (the idea that arterial elasticity has a buffering effect on the pulsatile nature of blood flow), e.g. [16], [18], [20]. Another modern approach, that is influential in CV modelling today, makes use of hydrodynamic pulse-wave models [6], [10], [16], [18]. Furthermore, a distinction can be made between so-called transmission line models [27], segmental models [7], [15], [23], [27] and hybrid models. What all these

approaches have in common is that they use rather complex models based on differential equations at the level of detailed physiology to describe the dynamics of this system.

In contrast, the current paper shows that the organisation modelling approach, although initially meant for purely social systems, provides adequate models in this type of application area as well. Realisers of roles within such an organisation models are active components of the biological system. As a result, this kind of biological organisations can also be considered in a way as (pseudo-)social systems, especially in the sense that the processes involved within these active components have to co-operate in a well-organised manner in order to produce the desired or required behavior for the overall system.

In Section 2 a brief introduction of the AGR organisation modelling approach can be found and illustrated for the context of the circulatory system. In Section 3 the dynamic properties at different levels of aggregation of this organisation model are identified. In Section 4 the relationships between these dynamic properties at different levels are presented. Section 5 describes how part of the dynamic properties can be used to enable a simulation of the circulatory system. In Section 6 the remaining properties are validated against the simulation of Section 5. Finally, Section 7 provides a description of how specific agents can be allocated to roles within the AGR approach.

2. The Organisation Structure of the Circulatory System

This section presents the organisation structure for the biological case study undertaken to investigate the usefulness of the AGR multi-agent organisation modelling approach to biological systems: the circulatory system in mammals. After a description of the functioning of the circulatory system, the AGR approach is briefly introduced. Next, the approach is applied to the circulatory system by identifying the organisational structure, expressed by AGR in terms of roles, groups, and interactions between these elements, and the agents realising these roles.

2.1. The Circulatory System

The circulatory system takes care for a number of capacities, such as providing nutrients and oxygen to the body and taking wastes (e.g., CO_2) out of the body; e.g., [18], [20]. The main property to focus on in this example is that the system provides oxygen for all parts of the body. The organisation of the circulatory system S is analysed as consisting of the following active components (or agents) that by showing their reactive and pro-active behavior all play their roles within the overall process:

- heart
- capillaries in lungs and other organs
- arteries
    - pulmonary artery channels (from the heart to the capillaries in the lungs)
    - aorta channels (from heart to the capillaries in the body)
- veins
    - pulmonary veins (from the capillaries in the lungs to the heart)
    - inferior and superior vena cava (from the capillaries in the body to the heart)

These active components work together due to some structure, as schematically depicted in Figure 1. Note that Figure 1 only describes the material structure of the circulatory system; the components depicted are physical components. Such pictures do not account for the role that the different physical components play in the organised process as a whole. For example the similarity in roles of the components in the systemic cycle (left hand side) and in the pulmonary cycle (right hand side) are not made precise. To clarify such functional and organisational aspects and similarities, the organisational structure will be described in the next subsections.

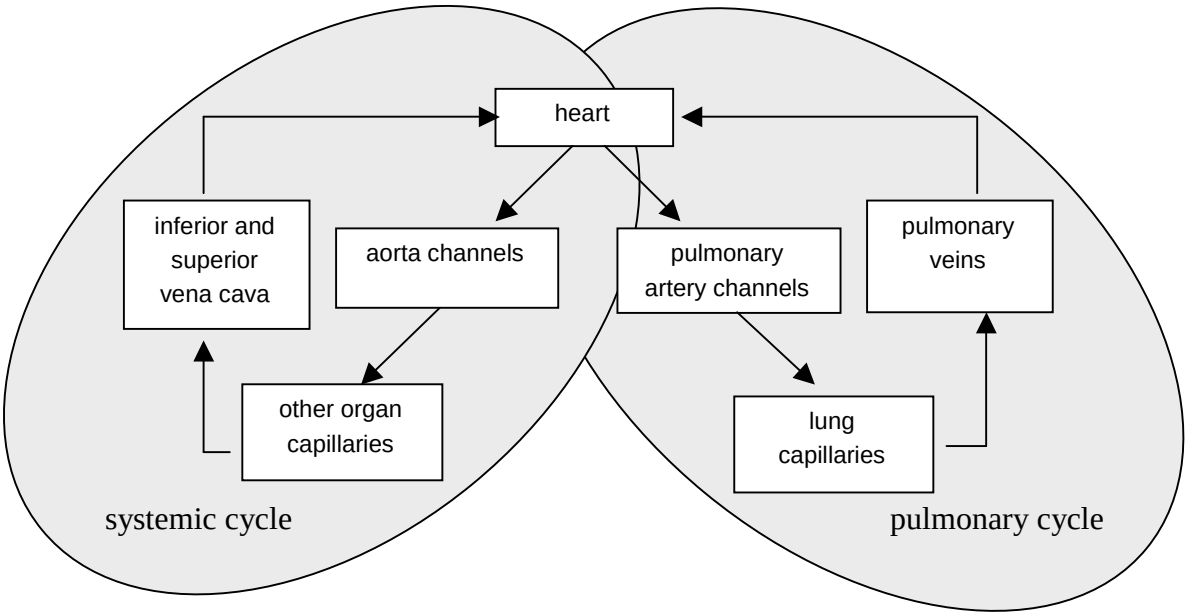


Figure 1. Schema for the circulatory system

2.2. AGR Organisational Structures

To model an organisation, the Agent/Group/Role (AGR) approach, adopted from [3] is used. The *organisational structure* is the specification of a specific multi-agent organisation based on a definition of groups, roles and their relationships within the organisation:

- An organisation as a whole is composed of a number of *groups*.
- A group structure identifies the *roles* and (*intragroup*) *interaction between roles*, and *transfers* between roles needed for such interactions.
- In addition, *intergroup* role relations between roles of different groups specify the connectivity of groups within an organisation.

The modelling approach is further explained and illustrated by the application to the circulatory system in mammals.

2.3. Groups and Roles within the Circulatory System

The left-hand side and the right-hand side of the picture in Figure 1 are organised according to a similar structure:

- The *heart* initiates the flow,
- which is led by (aorta, resp. pulmonary artery) *arteries* or *channels* to
- *organs* (lung, resp. other organs) where exchange takes place,
- from where the flow is led by (pulmonary, resp. inferior and superior vena cava) *veins*
- back to the *heart*.

Here, in each of the two sides the heart plays two roles, one of a well, initiating the flow, and one of a drain, where the flow disappears (and will re-appear in the other side).

The similarity of the two parts of the circulatory system enables to model their common structure in an abstract manner in the form of a more generic *group structure* G which has two instantiations within the circulatory system: one for the left hand side (called *systemic cycle*, used for oxygen supply, among others), and one for the right hand side (called *pulmonary cycle*, used for oxygen uptake, among others). Modelling the system from this perspective provides several advantages over the material perspective shown in Figure 1. For instance, the possibility to describe both main cycles by a single, generic group structure allows us to identify certain similarities between the two cycles. Moreover, such generic structures could enable comparative studies with systems in other organisms than mammals.

Generic Group Structure G

The generic group structure G (see Figure 2) consists of the following five *roles*: *well, supply guidance, exchange, drain guidance, drain*.

Transfers and intragroup role interactions within G

The transfers underlying the interactions between roles are depicted in Figure 2. A short explanation of these interactions is as follows:

well – supply guidance role interaction

If the well comes up with a new flow, then this flow will be picked up by the supply guidance, and transported further.

supply guidance – exchange role interaction

If the supply guidance delivers a flow, then the exchange role will take out substances from this flow and will insert other substances in the flow.

exchange – drain guidance role interaction

The flow resulting from the exchange will be picked up by and transported by the drain guidance.

drain guidance – drain role interaction

If the drain guidance delivers a flow, then this is picked up by the drain (which lets it disappear).

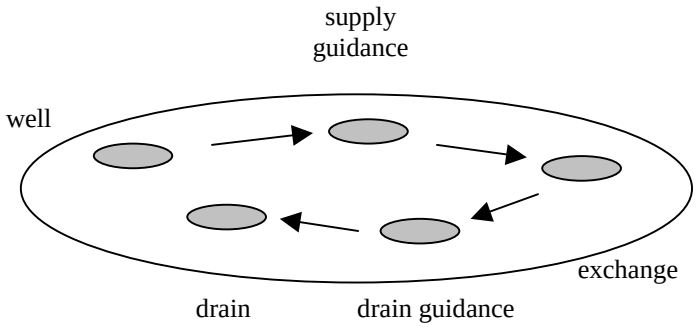


Figure 2. Roles and transfers within the generic group structure G

Group instances and role instances
Two instances of the generic group structure G are used: the *pulmonary cycle group instance* G_p and the *systemic cycle group instance* G_s. Based on the generic group structure G, for each of the group instances different role instances are defined. These role instances are denoted by using the group instance name as a prefix; i.e., the role instances *systemic cycle well, systemic cycle supply guidance, systemic cycle exchange, systemic cycle drain guidance, systemic cycle drain* within the systemic cycle group instance, and similar for the pulmonary group instance.

Allocation of agents to role instances
The relation between Figure 2 and Figure 1 is such that to each role instance depicted in Figure 2, a specific agent is allocated in Figure 1. This is the case for both the pulmonary cycle group instance and the systemic cycle group instance. In particular, for the systemic cycle group instance the allocation of agents to role instances is as follows:

heart	- systemic cycle well
aorta channels	- systemic cycle supply guidance
organ capillaries	- systemic cycle exchange
inferior and superior vena cava	- systemic cycle drain guidance
heart	- systemic cycle drain

For the pulmonary cycle group instance the allocation of agents to role instances is as follows:

heart	- pulmonary cycle well
pulmonary channels	- pulmonary cycle supply guidance
lung capillaries	- pulmonary cycle exchange
pulmonary veins	- pulmonary cycle drain guidance
heart	- pulmonary cycle drain

The allocation of agents to role instances is discussed in more detail in Section 7.

2.4. Connectivity between groups: intergroup role interactions

The connectivity between the groups within the organisation structure is realised by two intergroup role interactions: from the drain role instance within one group to the well role instance in the other group, in both directions; see Figure 3.

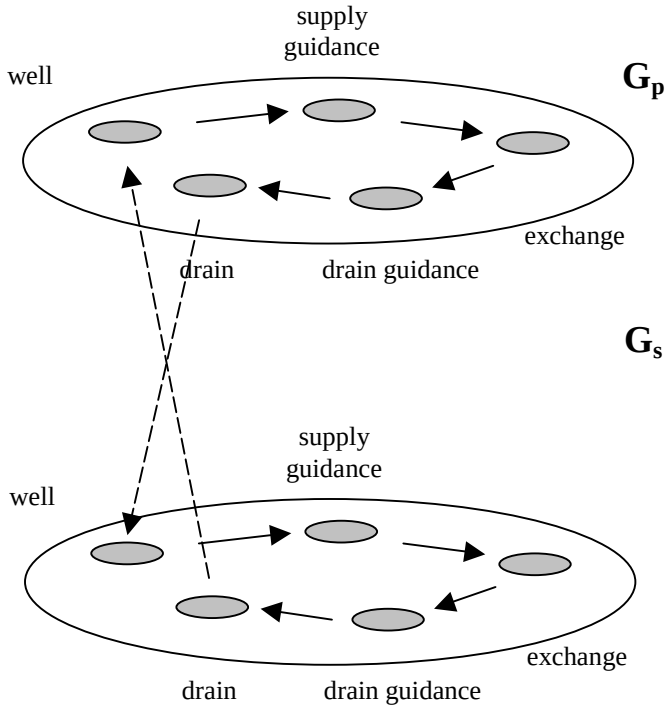


Figure 3. Intergroup role interactions

In a generic sense such an intergroup role interaction can be explained by stating that the flow taken out by the drain role instance in one group instance is supplied within the other group instance by the well role instance. For the two group instances in the example these interactions are briefly explained as follows.

pulmonary cycle drain – systemic cycle well role interaction

The oxygen-rich blood flow taken out by the pulmonary cycle drain role instance within the pulmonary cycle group instance is supplied to the systemic cycle well role instance within the systemic cycle group instance

systemic cycle drain – pulmonary cycle well role interaction

The oxygen-poor blood flow taken out by the systemic cycle drain role instance within the systemic cycle group instance is supplied to the pulmonary cycle well role instance within the pulmonary cycle group instance

3. Dynamic Properties at Different Levels within the Organisation

To describe the functioning of the circulatory system S as an organisation, the following types of dynamic properties can be used (in the paper limited to properties related to oxygen supply which is a core function of the circulatory system):

- dynamic properties of the organisation as a whole
- dynamic properties for groups and intergroup role interactions
- properties of roles, transfer properties and intragroup role interactions within a group.

Moreover, usually some environmental assumptions are needed. The argument "s" when appearing in the name of a property refers to the instance of that property suitable for the systemic cycle group, similarly the argument "p" refers to the pulmonary cycle group.

3.1. Environment Assumptions

For the circulatory system S two reasonable environmental assumptions are:

EA1 Oxygen availability
At any point in time oxygen is present in the lungs

EA2(i) Stimulus occurrence (with maximal interval i)
For any point in time t there exists a time point with $t < t' \leq t + i$ such that at t' a stimulus occurs.

3.2. Dynamic Properties of the Organisation as a Whole

Global properties can be expressed for proper functioning of the flow through the cycles (taken at the well), and for resulting oxygen provision through the capillaries.

GP1(w) Well successfulness (with maximal interval w)
After an initiation time t0, for any point t there exists a time point t' with $t < t' \leq t + w$ such that at t' a fluid with ingredients I is generated by the well.

Here I is a specification of ingredients, for example by a list of them, possibly with indications of concentrations.

Note that this global property depends on the organisation as a whole, not only on the group of the well. This property can be instantiated both for the well within the pulmonary cycle group ($GP1(p, w_p)$), and for the well within the systemic cycle group ($GP1(s, w_s)$).

GP2(d) Oxygen delivery successfulness (with maximal interval d)
After an initiation time t0, for any point t there exists a time point t' with $t < t' \leq t + d$ such that at t' by exchange oxygen is delivered to the organs.

3.3. Intergroup Role Interaction Properties

Intergroup role interaction properties relate roles in different groups. They typically express a dynamic relation between the input of one role in one group to the output of another role in another group. For the organisation of the circulatory system S consisting of two group instances as depicted in Figure 3 the following intergroup role interaction property has been specified. Again, this property can be instantiated both for the well within the pulmonary cycle group ($IrRI(p, c_p, r_p)$), and for the well within the systemic cycle group ($IrRI(s, c_s, r_s)$).

IrRI(c, r) Drain– well intergroup role interaction
At any point in time t0
if at some $t \leq t0$ the drain within some group instance G_i received a fluid volume V with ingredients I
and between t and t0 no stimulus occurred
and at t0 a stimulus occurs
then there exists a time point t1 with $t0 + c \leq t1 \leq t0 + r$ such that at t1 the well within the other group instance G_j generates a fluid volume V with ingredients I

3.4. Dynamic Properties of Groups

Within an overall organisation, each group's contribution can be formulated in the form of some group property. An example of such a group property is the following.

GR(u, v, u', v') Group successfulness
At any point in time t,
if at t the well generates a fluid volume V with ingredients I
then there exist time points $t' \leq t''$ with $t + u \leq t' \leq t + v$ and $t + u' \leq t'' \leq t + v'$ such that
at t' ingredient A is added to the environment and ingredient B taken from the environment
and at t'' the drain receives a fluid volume V with ingredients I - A + B

Here V is an amount of fluid and I is a specification of ingredients, as before. The notation I - A + B is used for the specification of the ingredients of I except A and augmented by B. The group specific property instances according to group instances are called GR(s, u_s, v_s, u'_s, v'_s) and GR(p, u_p, v_p, u'_p, v'_p). For the pulmonary group instance GR(p) the air is environment, A is carbonacid, and B is oxygen, for the systemic group instance GR(s) the environment is formed by the organs of the body, A is oxygen, and B is carbonacid. The difference in meaning of A and B for instantiations according to group instances is valid in other properties as well.

The dynamic properties of the different groups and of their interactions modelled by intergroup role interactions, contribute to the overall properties of S.

As discussed in [5], some dynamic group properties have a specific form in that they relate one role in the group to another role in the group. The two types of such properties that are relevant (transfer properties and intragroup role interaction properties) are discussed in the following section.

3.5. Transfer and Intragroup Role Interaction Properties

Intragroup role interaction properties characterise how roles (have to) interact. They typically relate the output of one role to the output of another role. This is slightly more abstract than role behavior and transfer properties.

IaRI(a1, b1) Well implies supply guidance
At any point in time t
if the well generates a fluid volume V with ingredients I
then there exists a time point t' with $t + a1 \leq t' \leq t + b1$ such that at t'
the supply guidance generates a fluid volume V with ingredients I

IaRI2(a2, b2) Supply guidance implies exchange
At any point in time t
if the supply guidance generates a fluid volume V with ingredients I
then there exists a time point t' with $t + a2 \leq t' \leq t + b2$ such that at t'
ingredient A is added to the object and ingredient B taken from the object
and the exchange generates a fluid volume V with ingredients I - A + B

IaRI3(a3, b3) Exchange implies drain guidance
At any point in time t
if the exchange generates a fluid volume V with ingredients J
then there exists a time point t' with $t + a3 \leq t' \leq t + b3$ such that at t'
the drain guidance generates a fluid volume V with ingredients J

Transfer properties express that the different roles are connected in an appropriate manner to enable proper interaction. For each of the four arrows in Figure 3 a transfer property expresses that the proper connection exists between the output of one role and the input of the other role. In a general form delays can be taken into account for the transfers. However, for this example, these delays for transfers are assumed to be 0 (input

state property is assumed identical to previous output state property), i.e., all gi's and hi's are 0.

TR1(g1, h1) Well connects to supply guidance
At any point in time t
if the well generates a fluid volume V with ingredients I
then there exists a time point t' with $t + g1 \leq t' \leq t + h1$ such that at t'
the supply guidance receives a fluid volume V with ingredients I

This property is not fulfilled, for example, if the well opening is not connected to the supply guidance, so that the generated fluid volume streams away in the environment without reaching the supply guidance.

TR2(g2, h2) Supply guidance connects to exchange
At any point in time t
if the supply guidance generates a fluid volume V with ingredients I
then there exists a time point t' with $t + g2 \leq t' \leq t + h2$ such that at t'
the exchange receives a fluid volume V with ingredients I

TR3(g3, h3) Exchange connects to drain guidance
At any point in time t
if the exchange generates a fluid volume V with ingredients I
then there exists a time point t' with $t + g3 \leq t' \leq t + h3$ such that at t'
the drain guidance receives a fluid volume V with ingredients I

TR4(g4, h4) Drain guidance connects to drain
At any point in time t
if the drain guidance generates a fluid volume V with ingredients I
then there exists a time point t' with $t + g4 \leq t' \leq t + h4$ such that at t'
the drain receives a fluid volume V with ingredients I

3.6. Role Behavior Properties

Role behavior properties abstract from the specific agent allocated to a role, but characterise which behavior an agent fulfilling this role needs to have. Such properties typically relate the input of a role to the output of the same role.

supply guidance behavior

The arteries contribute in transportation. This means that that if their input receives blood, then their output generates blood with the same ingredients.

RB1(e1, f1) Supply guidance effectiveness
At any point in time t
if the supply guidance receives a fluid volume V with ingredients I
then there exists a time point t' with $t + e1 \leq t' \leq t + f1$ such that at t'
it generates a fluid volume V with ingredients I

exchange behavior

RB2(e2, f2) Exchange effectiveness
At any point in time t
if the exchange receives a fluid volume V with ingredients I
then there exists a time point t' with $t + e2 \leq t' \leq t + f2$ such that at t'
ingredient A is added to the object (environment, i.e., lung or organ)
and ingredient B is taken from the object
and it generates a fluid volume V with ingredients I - A + B

drain guidance behavior

RB3(e3, f3) Drain guidance effectiveness
At any point in time t
if the drain guidance receives a fluid volume V with ingredients I
then there exists a time point t' with $t + e3 \leq t' \leq t + f3$ such that at t'
it generates a fluid volume V with ingredients I

4. Relationships between Dynamic Properties at Different Levels

The idea is that dynamics of the whole organised (multi-agent) system is generated by lower level properties, in particular by the group properties and intergroup interaction properties. In turn, group dynamics is generated by role behavior and transfer within a group. This is elaborated in more detail by identifying logical relationships between these dynamic properties.

4.1. Overall Properties: Oxygen Delivery Successfulness

The global property GP2 (oxygen delivery successfulness) depends on the systemic cycle instance of global property GP1 (well successfulness), assuming proper group functioning of the same group instance. To be more precise, the following relationship holds:

$$GP1(s, w) \;\&\; GR(s, u_s, v_s, u'_s, v'_s) \;\Rightarrow\; GP2(d)$$
with $d = w + v_s$

So property GP2(d) is implied by two other properties, i.e., GP1(s, w) and GR(s, u_s, v_s, u'_s, v'_s). This implication are depicted in Figure 4. A sketch of a proof of this implication is as follows. Suppose GP1(s, w) holds. Then, after an initiation time t0, for any point t there exists a time point t' with $t < t' \leq t + w$ such that at t' a fluid with ingredients I is generated by the well of the systemic cycle. And if GR(s, u_s, v_s, u'_s, v'_s) holds as well, this means that the systemic cycle works correctly. Thus, from the fluid generated by the well, oxygen is finally taken and delivered to the organs. It can be concluded that after an initiation time t0, for any point t there exists a time point t' with $t < t' \leq t + d$ such that at t' by exchange oxygen is delivered to the organs, which is exactly what GP2(d) states. Furthermore, it is known that w is the maximum time interval for fluid generation by the well, and v_s is the maximum time interval for oxygen supply by the systemic cycle. Hence, it follows logically that $d = w + v_s$.

The relationships that GP1(s, w) and GR(s, u_s, v_s, u'_s, v'_s) have with other properties are depicted in Figures 5 and 6.

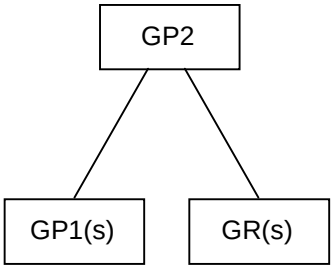


Figure 4. Oxygen delivery successfulness related to global property GP1(s) and a group property.

4.2. Overall Properties: Well Successfulness

Well successfulness depends on proper functioning of the whole cycle; it needs as input that a fluid volume is received. If the whole cycle functions well, the group properties, intergroup role interaction properties, and environmental assumption EA2 guarantee that this well functioning is maintained. However, the process needs a starting point. This starting point is assumed for the well within both groups at time point t = 0 in the following form:

Init(w_{init}) **Well initialisation**
There exists a time point t with $0 \leq t \leq w_{init}$ such that at t
the well in the pulmonary group instance generates a fluid volume V with any ingredients I
and the well in the systemic group instance generates a fluid volume V' with any ingredients I'

Using these properties the following relationships can be established (see also Figure 5).

$$\text{Init}(w_{init}) \,\&\, \text{GR}(s, u_s,v_s,u'_s,v'_s) \,\&\, \text{GR}(p, u_p,v_p,u'_p,v'_p) \,\&\, \text{IrRI}(s, c_s,r_s) \,\&\, \text{IrRI}(p, c_p,r_p) \,\&\, \text{EA2}(i) \Rightarrow \text{GP1}(s, w_s)$$
with $w_s = \max(w_{init}, \max(i, v'_p)+r_s)$

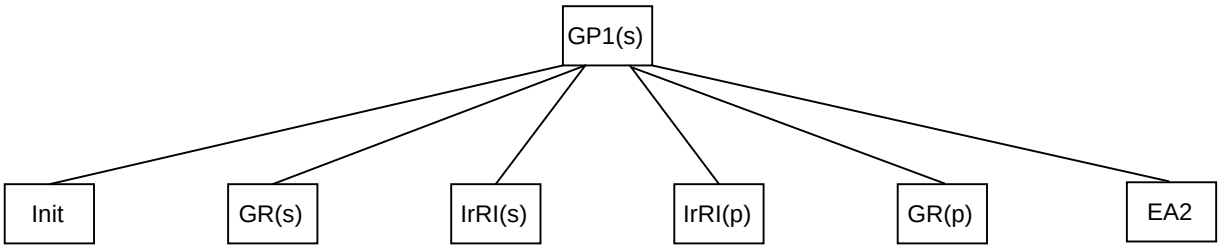


Figure 5. Global property GP1(s) related to other properties

4.3. Group Properties

A group property is related in an integrative manner to a combination of intragroup role interaction properties.

$$\text{IaRI1}(s, a1_s, b1_s) \,\&\, \text{IaRI2}(s, a2_s, b2_s) \,\&\, \text{IaRI2}(s, a3_s, b3_s) \Rightarrow \text{GR}(s, u_s,v_s,u'_s,v'_s)$$
with $u_s = a1_s + a2_s$, $v_s = b1_s + b2_s$, $u'_s = a1_s + a2_s + a3_s$, $v'_s = b1_s + b2_s + b3_s$

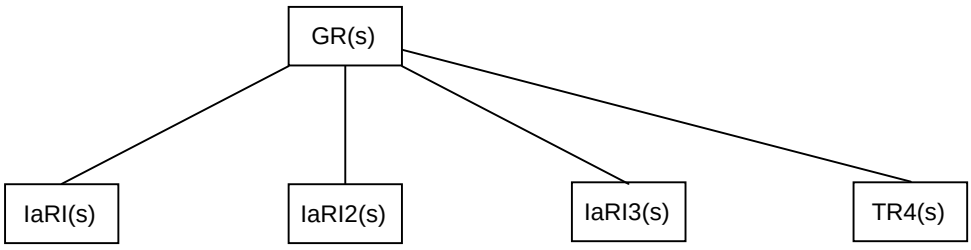


Figure 6. Group property related to intragroup interaction properties

Intragroup role interaction properties relate to role behavior properties and transfer properties in the following manner.

$$\text{TR1}(s) \,\&\, \text{RB1}_s(s, e1_s, f1_s) \Rightarrow \text{IaRI1}(s, e1_s, f1_s)$$
$$\text{TR2}(s) \,\&\, \text{RB2}(s, e2_s, f2_s) \Rightarrow \text{IaRI2}(s, e2_s, f2_s)$$

$$TR3(s) \ \& \ RB3(s, e3_s, f3_s) \ \Rightarrow \ IaRI3(s, e3_s, f3_s)$$

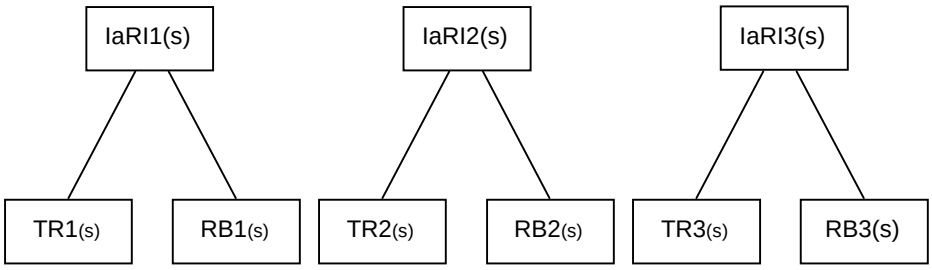


Figure 7. Intragroup interaction properties related to role behavior and transfer properties

4.4. Overview

In Figure 8 an overview can be found for all dynamic properties relating to $GP1_s$.

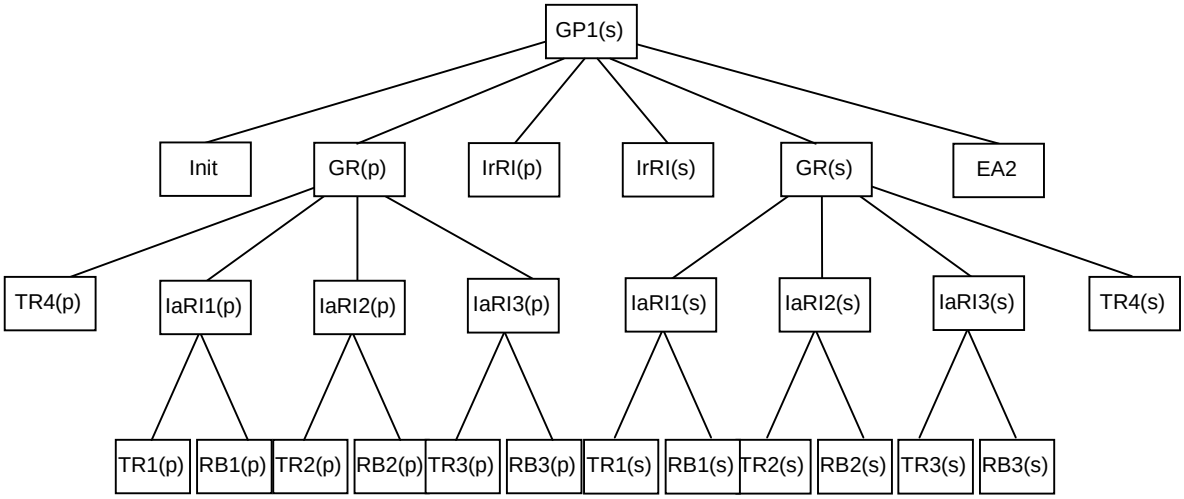


Figure 8. Overview of the interlevel relationships for global property GP1(s)

5. Simulation

A software environment has been created to enable the simulation of executable organisation models specified at a high conceptual level [10]. The input of this simulation environment is a set of dynamic properties in a specific, executable format. In [9] the language TTL was introduced as an expressive language for the purpose of specification and checking of dynamic properties. For the purpose of simulation, to obtain computational efficiency the format used for dynamic properties is more restricted than the TTL format used to specify various types of dynamic properties: they are in so-called *leads to* format; cf. [10]. This is a real time-valued variant of Executable Temporal Logic [1]. Roughly spoken, in *leads to* format the following can be expressed:

if a state property α holds for a time interval with duration g, then after some delay (between e and f) another state property β will hold for a time interval h

This specific temporal relationship *leads to* is applicable forward as well as backward in time. Hence, if α and β are state properties, and α leads to β, this also means that if β

holds for a time interval of length h, then α held during some time interval with length g, of which the starting point was between e and f before the starting point of the second interval. A formal definition of this *leads to* relation is as follows. Here state($\mathcal{T}$, t) denotes the state at time t in trace $\mathcal{T}$, and S $\models$ α that in a state S state property α holds. Moreover, Traces denotes the set of all possible traces.

Definition

(a) Let α , β ∈ SPROP(AllOnt). The state property α *follows* state property β, denoted by $\alpha \rightarrow\!\!\!\!\rightarrow_{e, f, g, h} \beta$, with time delay interval [e, f] and duration parameters g and h if:
$\forall \mathcal{T} \in$ Traces ∀t1:
[∀t ∈ [t1 - g, t1) : state($\mathcal{T}$, t) $\models$ α ⇒ ∃d ∈ [e, f] ∀t ∈ [t1 + d, t1 + d + h) : state($\mathcal{T}$, t) $\models$ β]

(b) Conversely, the state property β *originates from* state property α, denoted by $\alpha \bullet\!\!-_{e, f, g, h} \beta$, with time delay in [e, f] and duration parameters g and h if:
$\forall \mathcal{T} \in$ Traces ∀ t2:
[∀t ∈ [t2, t2 + h) : state($\mathcal{T}$, t) $\models$ α ⇒ ∃d ∈ [e, f] ∀t ∈ [t2 - d - g, t2 - d) state($\mathcal{T}$, t) $\models$ β]

(c) If both $\alpha \rightarrow\!\!\!\!\rightarrow_{e,f,g,h} \beta$, and $\alpha \bullet\!\!-_{e,f,g,h} \beta$ hold, then α *leads to* β, denoted by:
$\alpha \bullet\!\!\rightarrow\!\!\!\!\rightarrow_{e,f,g,h} \beta$.

Making use of these *leads to* properties, the software environment generates simulation traces (actually the *follows* relations are used in the simulation software; if in a specification there is only one way to reach each β, then this automatically results in *leads to* relations holding). A trace is developed by starting at time t = 0 and for each time point up to which the trace already has been constructed, checking which antecedents of executable properties hold in the already constructed trace. For these executable properties, add the consequent to the trace, i.e., extend the trace in time in such a manner that the consequent holds.

The relation between the specification and the constructed trace is that the trace is a model (in the logical sense) of the theory defined by the specification, i.e., all executable dynamic *leads to* properties of the specification hold in the trace.

To be able to simulate the behavior of the circulatory system, all leaves of the tree in Figure 8 have been expressed in *leads to* format. That is, all intergroup role interaction properties, role behavior properties, transfer properties, and the special starting point property Init. The values chosen for the timing parameters are shown in Table 1.

Property	**Minimal delay (e)**	**Maximal delay (f)**	**Duration antecedent (g)**	**Duration consequent (h)**
RB1(p)	3	5	0	0
RB1(s)	10	20	0	0
RB2(p)	5	10	0	0
RB2(s)	5	10	0	0
RB3(p)	3	5	0	0
RB3 (s)	10	20	0	0
IrRI(p)	5	10	1	10
IrRI(s)	5	10	1	10

Table 1. Time parameters for *leads to* properties

The resulting trace is shown in Figure 9. Time is on the horizontal axis, the properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false during that time period. The line labeled *stimulus_occurs*, for example, depicts the property that a heart stimulus occurs. This property is true from time point 0 to 5, from 80 to 85, from 160 to 165, and so on. Notice that this is exactly the intended dynamics according to environmental assumption EA2. Also notice that for the maximum interval s within EA2, the value 80 has been chosen within this example. Furthermore, Figure 9 shows that after a stimulus has occurred, the wells of both groups generate fluid, which is immediately received by the supply guidances (since the delays for transfers were assumed to be 0). After that, in both groups the fluid continues to the exchange. Since the systemic cycle is longer than the pulmonary cycle (the aorta channels are longer than the pulmonary artery channels), it takes more time for the supply guidance in the systemic group to generate fluid. Next, some moments after the exchange has received a fluid, it can be seen that the ingredients are actually exchanged. After that, fluid goes from the exchange to the drain guidance and finally to the drain.

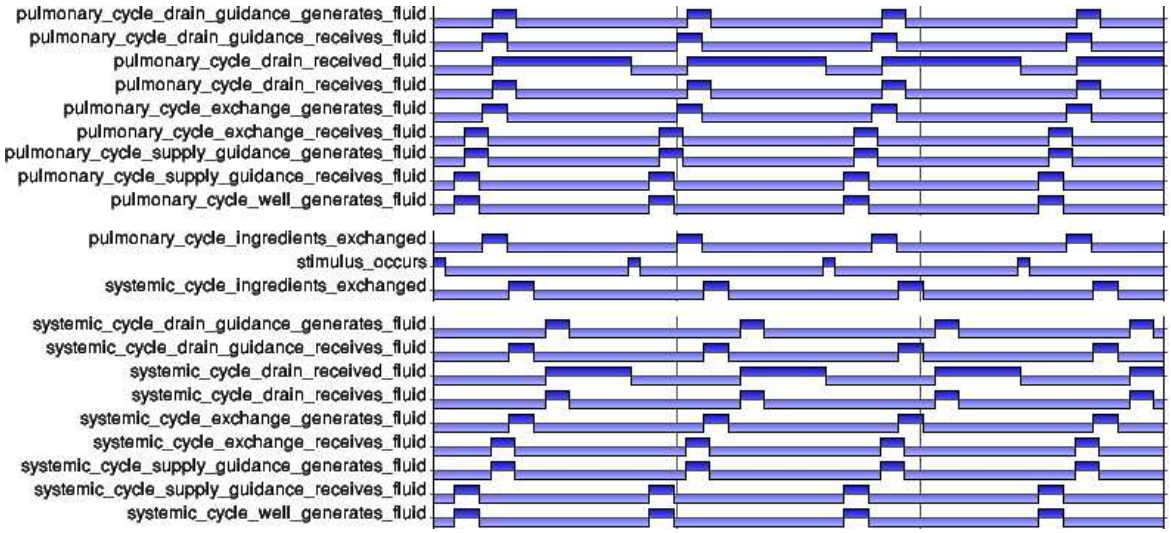


Figure 9. Results of the simulation of executable properties of the circulatory system

6. Checking Properties

Logical relationships between properties, as depicted in the tree of Figure 8, can be very useful in the analysis of dynamic properties of an organisation (like the circulatory system in this particular case); also see [8]. For example, if for a given trace of the system the global property GP1(s) is not satisfied, then by a refutation process it can be concluded that either one of the group properties, or one of the intergroup role interaction properties, or the property Init does not hold. If, after checking these properties, it turns out that GR(p) does not hold, then either one of the intragroup role interaction properties or TR4(p) does not hold. By this refutation analysis it follows that if GP1(s) does not hold for a given trace, then, via the intermediate properties, the cause of this malfunctioning can be found in the set of leaves of the tree of Figure 8.

In order to determine which one of the properties encountered in this refutation process actually is refuted, some mechanism is needed to check if a certain property holds for a given trace. To this end, the simulation software described above automatically produces

log files containing the traces. In addition, software has been developed that is able to read in these log files together with a set of dynamic properties (in *leads to* format), and to perform the checking process. This is done in two directions. On the one hand, each atom occurring in the trace is 'explained', i.e., the software verifies if there was a reason for its presence, according to the dynamic properties. On the other hand, for each atom a check is performed whether all atoms it implies according to the dynamic properties are actually there. As a result, the software determines not only whether the properties hold for the trace or not, but in case of failure, it also pinpoints which parts of the trace violate the properties. If a property does not hold completely, this is marked by the program. Yellow marks indicate unexpected events, occurring when certain atoms cannot be explained. Red marks indicate events that have not happened, whilst they should have happened. Checks of this kind have actually been performed for all of the higher level properties of Figure 8, i.e., for all nodes of the tree that are no leaves. They all turned out to hold for the trace of Figure 9, which validates the tree.

In addition, recently other software has been developed (and is still being improved) that is able to check traces against properties in the *TTL* format instead of the *leads to* format. Since TTL, as mentioned in Section 5, has a considerably higher expressiveness, this new software enables to check much more complex properties. For instance, for the present case study, the property "*the higher the number of stimuli, the more oxygen is delivered in the lungs*" has been checked successfully. Checks of this kind are normally performed in less than a second. Future work involves exploring the limits to the amount of complexity that the software can handle.

7. Realisation of the Organisation by Allocation of Agents

An organisation model such as the one presented in this paper provides an abstract model for the manner in which multiple interacting processes or agents generate dynamics. The specific agents are not part of such an organisation model. Instead the notion of role provides an abstract entity or placeholder for where specific agents come in. In the example domain addressed here these agents are active biological components such as the heart, lungs, and other organs. An important advantage of this abstraction is that the dynamics can be modeled independent of the specific choices of agents. The organisation model can be (re)used for any allocation of agents to roles for which:

- for each role, the allocated agent's behavior satisfies the dynamic role properties,
- for each intergroup role interaction, one agent is allocated to both roles and its behavior satisfies the intergroup role interaction properties, and
- the communication between agents satisfies the respective transfer properties.

Expressed differently, for a given allocation of agents to roles the following logical relationships between dynamic properties hold:

agent – role
from dynamic agent properties to dynamic role properties:

agent A is allocated to role r &
dynamic properties of agent A $\Rightarrow$
dynamic properties of role r

As an example for the case of the circulatory system, one can think of the aorta channels as agent A and of the systemic cycle supply guidance as role r (also see the allocation schema at the end of Section 2.3).

agent – intergroup role interaction
from dynamic agent properties to dynamic intergroup role interaction properties:

agent A is allocated to roles r1 and r2 in different groups &
dynamic properties of agent A ⇒
dynamic properties of intergroup role interaction between r1 and r2

As an example, one can think of the heart as agent A and of the systemic cycle well and the pulmonary cycle drain as role r1 and r2, respectively.

agent communication – role transfer
from dynamic agent communication properties to dynamic transfer properties:

agent A is allocated to role r1 and agent B to role r2 in one group &
dynamic properties of communication from A to B ⇒
dynamic properties of transfer from r1 to r2

As an example, one can think of the aorta channels as agent A, of the systemic cycle supply guidance as role r1, of the organ capillaries as agent B and of the systemic cycle exchange as role r2.

8. Discussion

The aim of this paper was to investigate whether modelling techniques from the area of organisation modelling (already shown to be successful for human organisations in, e.g., [8], [11]) provide adequate means to model at a high level of abstraction the dynamics of biological systems in which multiple distributed interacting processes play a role. As a case study the circulatory system in biological organisms (mammals) was explored using a chosen organisation modelling framework.

In the literature, many different kinds of cardiovascular models exist, typically based on modelling the physiology by differential equations. In contrast to these mathematical models of the circulatory system our paper shows how an organisation modelling approach such as the chosen one (other organisation modelling approaches may well be as applicable as the chosen one) can be used to model the dynamics of biological organisation for the case of the circulatory system at a high conceptual level. This system consists of a number of components that are connected and grouped together in such a manner that everything functions in a coherent manner. It was shown how active components within the circulatory system can be considered realisers of the roles within the organisation model. Dynamic properties at different levels of aggregation of this organisation model have been identified, and logical interlevel relationships between these dynamic properties at different aggregation levels were made explicit. Based on the executable properties, simulation has been performed and properties have been (automatically) checked for the produced simulation traces. Thus the logical interlevel relationships between properties have been verified. The variant of executable temporal logic (extending the approach described in [1]) used for simulation has as an advantage that it is guaranteed that a generated trace satisfies the specified executable dynamic properties. Since these dynamic properties stand in logical relationships to other (more complex, not necessarily executable) dynamic properties, this form of simulation facilitates logical analysis of the dynamics at different levels of aggregation.

In summary, it turned out that, at least for the chosen domain, the chosen organisation modelling approach provides adequate means for high-level modelling of the complexity of the dynamics of biological organisms. For example, a strong contrast in abstraction and

manageability of the model was found with modelling techniques based on differential equations that provide less transparent, low-level models. This outcome was confirmed by a case study in another biological domain in which the organisation of intracellular processes was modelled.

References

[1] Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M., (1996). The Imperative Future: Principles of Executable Temporal Logic, Research Studies Press Ltd. and John Wiley & Sons.

[2] Ferber, J., (1999). *Multiagent Systems*. Addison Wesley.

[3] Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organisations in multi-agent systems. In: *Proc. of the Third International Conference on Multi-Agent Systems* (ICMAS'98), IEEE Computer Society Press, pp. 128-135.

[4] Ferber, J., and Gutknecht, O. (1999). Operational Semantics of a role-based agent architecture. *Proceedings of the 6th Int. Workshop on Agent Theories, Architectures and Languages (ATAL'1999)*. In: Jennings, N.R. & Lesperance, Y. (eds.) *Intelligent Agents VI*, Lecture Notes in AI, vol. 1757, Springer Verlag, 2000, pp. 205-217.

[5] Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., (2001). Organization Models and Behavioural Requirements Specification for Multi-Agent Systems. In: Y. Demazeau, F. Garijo (eds.), *Multi-Agent System Organisations. Proceedings of the 10th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'01*, 2001. Lecture Notes in AI, Springer Verlag. To appear, 2002.

[6] Fung, Y.C. (1984). Biodynamics: Circulation, New York, Springer-Verlag.

[7] Greenway, C.V. (1982). Mechanisms and quantitative assessment of drug effects on cardiac output with a new model of the circulation. Pharm Rev 33(4):213.

[8] Jonker, C.M., Letia, I.A., and Treur, J., (2002). Diagnosis of the Dynamics within an Organisation by Trace Checking of Behavioural Requirements. In: Wooldridge, M., Weiss, G., and Ciancarini, P. (eds.), *Proc. of the 2nd International Workshop on Agent-Oriented Software Engineering, AOSE'01*. Lecture Notes in Computer Science, vol. 2222. Springer Verlag, 2002, pp. 17-32.

[9] Jonker, C.M. and Treur, J. (1998). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), Proceedings of the International Workshop on Compositionality, COMPOS'97. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380. Extended version in: *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

[10] Jonker, C.M., and Treur, J., Relating Structure and Dynamics in an Organisation Model. In: J.S. Sichman, F. Bousquet, and P. Davidson (eds.), *Multi-Agent-Based Simulation II, Proc. of the Third International Workshop on Multi-Agent Based Simulation, MABS'02*, Lecture Notes in AI, vol. 2581, Springer Verlag, pp. 50-69.

[11] Jonker, C.M., Treur, J., and Wijngaards, W.C.A. (2002). Temporal Languages for Simulation and Analysis of the Dynamics Within an Organisation. In: B. Dunin-Keplicz and E. Nawarecki (eds.), *From Theory to Practice in Multi-Agent Systems, Proc. of the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems, CEEMAS'01*, 2001. Lecture Notes in AI, vol. 2296, Springer Verlag, 2002, pp. 151-160.

[12] Kreitner, R., and Kunicki, A. (2001). Organisational Behavior, McGraw – Hill.Li, J.K.-J. (1987). Arterial System Dynamics, NewYork, New York University Press.

[13] Lomi, A., and Larsen, E.R. (2001). Dynamics of Organizations: Computational Modeling and Organization Theories, AAAI Press, Menlo Park.

[14] Manna, Z., and Pnueli, A. (1995). *Temporal Verification of Reactive Systems: Safety*. Springer Verlag.

[15] Martin, J.F., Schneider, A.M., Mandel, J.E., et al. (1986). A new cardiovascular model for real-time applications. Trans Soc Comp Sim 3(1):31.

[16] Milnor, W.R. (1989). Hemodynamics 2nd ed, Baltimore, Williams & Wilkins.
[17] Mintzberg, H. (1979). The Structuring of Organisations, Prentice Hall, Englewood Cliffs, N.J.
[18] Noordergraaf, A. (1978). Circulatory System Dynamics. Academic Press, New York.
[19] Prietula, M., Gasser, L., Carley, K. (1997). Simulating Organizations. MIT Press.
[20] Rideout, V.C. (1991). Mathematical and Computer Modelling of Physiological Systems. Prentice Hall, Englewood Cliffs.
[21] Rizzi, M., Baltes, M., Theobald, U. & Reuss, M. (1997). *In vivo* analysis of metabolic dynamics in *Saccharomyces cerevisiae*: II. Mathematical model. *Biotechnol. Bioeng.* 55, 592–608.
[22] Rohwer, J.M., Meadow, N.D., Roseman, S., Westerhoff, H.V., Postma, P.W. (2000). Understanding glucose transport by the bacterial phosphoenolpyruvate:glycose phosphotransferase system on the basis of kinetic measurements in vitro. J. Biol. Chem., 275(45):34909-21.
[23] Slate, J.B., Sheppard, L.C. (1982). Automatic control of blood pressure by drug infusion. IEE Proc., Pt. A 129(9):639.
[24] Wang, J., Gilles, E.D., Lengeler, J.W., and Jahreis, K. (2001). Modeling of inducer exclusion and catabolite repression based on a PTS-dependent sucrose and non-PTS-dependent glycerol transport systems in Escherichia Coli K-12 and its experimental verification. *J. Biotechnol.* 2001, Dec 28; 92(2): 133-58.
[25] Weiss, G. (ed.) (1999). *Multiagent Systems*. MIT Press
[26] Westerhoff, H.V. (2001) The silicon cell, not dead but live! Metab. Eng. 2001; 3(3):207-10.
[27] Westerhof, N., Noordergraaf, A. (1969). Reduced models of the systemic arteries. Proc 8th Int Conf Med Eng, Chicago.
[28] Yu, C., Roy, R.J., Kaufman, H. (1990). A circulatory model for combined nitroprusside-dopamine therapy in acute heart failure. Med Prog Tech 16:77.

PART V SINGLE VS. MULTIPLE AGENTS

CHAPTER 15

Modelling the Dynamics of Complex Biological Processes as an Organisation of Multiple Agents

Part of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2005). Modelling the Dynamics of Intracellular Processes as an Organisation of Multiple Agents. In: Armano, G., Merelli, E., Denzinger, J., Martin, A., Miles, S., Tianfield, H., and Unland, R. (eds.), *Proceedings of the First International Workshop on Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics, MAS*BIOMED'05*, pp. 107-121.

Modelling the Dynamics of Complex Biological Processes as an Organisation of Multiple Agents

Tibor Bosse[1], Catholijn M. Jonker[1,2], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email : {tbosse, jonker, treur}@cs.vu.nl
URL : http://www.cs.vu.nl/~{tbosse, jonker, treur}
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands

Abstract. This paper explores how the dynamics of complex biological processes can be modeled as an organisation of multiple agents. This modelling perspective identifies organisational structure occurring in complex decentralised processes and handles complexity of the analysis of the dynamics by structuring these dynamics according to an organisational structure. More specifically, dynamic properties at different levels of aggregation in the organisational structure are identified, and related to each other according to the organisational structure. The applicability of this organisational modelling approach to address complexity in biological context is illustrated by a case study: the organisation of intracellular processes.

1. Introduction

To handle complex decentralised dynamics, often some type of organisational structure is exploited. The dynamics that emerge from multiple interacting agents within human society have been studied within Social Sciences in the area of Organisation Theory and within Artificial Intelligence in the area of Agent Systems; e.g., (Kreitner and Kunicki, 2001; Mintzberg, 1979; Lomi and Larsen, 2001; Prietula, Gasser and Carley, 1997; Ferber, 1999; Weiss, 1999). To manage complex, decentralised dynamics in human society, organisational structure is a crucial element: organisation provides a structuring and co-ordination of the processes in such a manner that a process or agent involved can function in a more adequate manner. The dynamics shown by a given organisational structure are much more dependable than in an entirely unstructured situation. It is assumed that the organisational structure itself is relatively stable, i.e., the structure may change, but the frequency and scale of change are assumed low compared to the more standard dynamics through the structure. Also in Nature several forms of organisational structure have been developed; typical examples are a beehive, the coordinated processes of organs in mammals, and the well-organised biochemistry of a living cell.

By using multi-agent organisation modelling techniques for analysis and simulation, the inherent complexity of the dynamics of multiple interacting processes within a society can be made manageable by choosing the right level of abstraction in describing them. As also discussed in (Jonker and Treur, 2005), many phenomena in Nature (or in the laboratory) have the same characteristic: they also involve complex dynamics of multiple distributed processes and their interaction. Therefore, a natural question is whether a multi-agent-organisation modelling perspective is promising for this domain of biological complexity.

Organisations can be viewed in two ways: (1) as adaptive complex information processing systems of (boundedly) rational agents, and (2) as tools for control; central issues are (Lomi and Larsen, 2001):

- How to identify properties of the whole, given properties of parts; from the first view: 'given a set of assumptions about (different forms of) individual behaviour, how can the aggregate properties of a system be determined (or predicted) that are generated by the repeated interaction among those individual units?'
- How to identify properties of parts, given desired or required properties of the whole; from the second view: 'given observable regularities in the behaviour of a composite system, which rules and procedures - if adopted by the individual units - induce and sustain these regularities?'

Recently a number of formal and computational modelling techniques have been developed that can be used for simulation or for formal analysis of the dynamics within a multi-agent organisation. Examples of this formalisation trend can be found in books such as (Prietula, Gasser, and Carley, 1997; Lomi and Larsen, 2001), and in a recently created journal: Computational and Mathematical Organisation Theory; e.g., (Moss *et al.*, 1998). For an organisation, different levels of aggregation can be identified, from single agent behaviour to the dynamics of the overall organisation. Dynamics can be described in an abstract manner by focusing on one of these levels and specifying dynamic properties for this level. Moreover, interlevel relationships between dynamic properties at different levels can be identified.

One of the organisation modelling approaches that have been developed within the agent systems area is the Agent-Group-Role (AGR) approach, introduced in (Ferber and Gutknecht, 1998), extended with operational semantics in (Ferber and Gutknecht, 2000), and with a modelling approach for dynamic properties in (Ferber, Gutknecht, Jonker, Müller and Treur, 2001). A related dynamic modelling framework for specification, analysis and simulation of AGR-organisation models, and supported by a software environment is described in (Jonker, Treur, and Wijngaards, 2002). This dynamic modelling environment allows to:

- *specify dynamic properties* for the different elements and levels of aggregation within an AGR organisation model
- *relate these dynamic properties* to each other according to the organisational structure
- use dynamic properties in *executable* form as a declarative *specification of a simulation model*
- perform *simulation experiments*
- automatically *check dynamic properties* for simulated or empirical traces

In this paper, first in Section 2 (Ferber and Gutknecht, 1998)'s Agent-Group-Role (AGR) organisation modelling approach is introduced, with an emphasis on organisational structure. It is illustrated by a model of the organisational structure of intracellular processes within *E.coli*. Section 3 addresses the dynamics of the organisation, described in terms of dynamics properties expressed in a Temporal Trace Language. In Section 4 relations between different levels of aggregation are discussed. Next, Section 5 provides some simulation results, and Section 6 concludes the paper by a discussion.

2. Organisational structure

One of the organisation modelling approaches that have been developed within the Agent Systems area is the Agent-Group-Role (AGR) approach (Ferber and Gutknecht, 1998). In this section, first a brief introduction of the AGR organisation modelling approach can be found (Section 2.1). In Section 2.2 the use of the approach is illustrated by describing the internal organisational structure of the unicellular organism *Escherichia coli* (Neidhardt, Curtiss III, Ingraham, Lin, Brooks Low, Magasanik, Reznikoff, Riley, Schaechter, and Umbarger, 1996). In this example, which for reasons of presentation is kept limited, the main property to focus on is growth under different environmental circumstances.

2.1. The AGR Organisation Modelling Approach

An AGR organisational structure for an overall process (or organisation) is a specification based on a definition of groups, roles and their relationships. An organisation as a whole is composed of a number of *groups*. A group structure identifies the *roles* and the *intragroup transfers* between roles. In addition, *intergroup role interactions* between roles of different groups specify the connectivity of groups within an organisation. *Agents* are allocated to roles; they realise the organisation. However, the aim of an organisation model is to abstract from any specific agent allocated. Therefore instead of particular agents, roles are used as abstract entities, defining properties agents should have when they are to function in a given role within an organisation. In Section 2.2 the AGR organisation modelling approach is illustrated for the unicellular organism *E. coli*.

2.2. Organisational structure of the living cell

In Figure 1 the aggregation levels of the AGR-organisation model of *E. coli* are depicted. In this picture the right hand side nodes connected to a node are called the children of the latter node, which itself is called a parent node for those children.

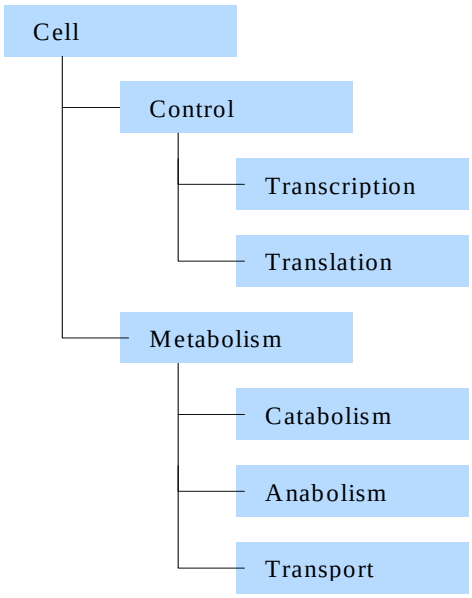


Figure 1. Overview of the aggregation levels of the organisation model of E.coli.

For example, the node Cell is the parent node of the nodes Control and Metabolism. The latter nodes are children of Cell. This means that they are the main categories or functional units that are distinguished for the processes in the cell. To be more specific, Metabolism and Control are the main parts of the regulation and control cycle of a cell. At one aggregation level lower, the Metabolism expands to Catabolism, Anabolism and Transport. The Catabolism is the category of processes that decompose substances and extract free energy from them. In the Anabolism the processes that utilise this free energy to create more and more complex substances reside. The Transport processes move substances across the cell membrane. The Control is decomposed into Transcription and Translation. These processes generate mRNA and enzymes, respectively.

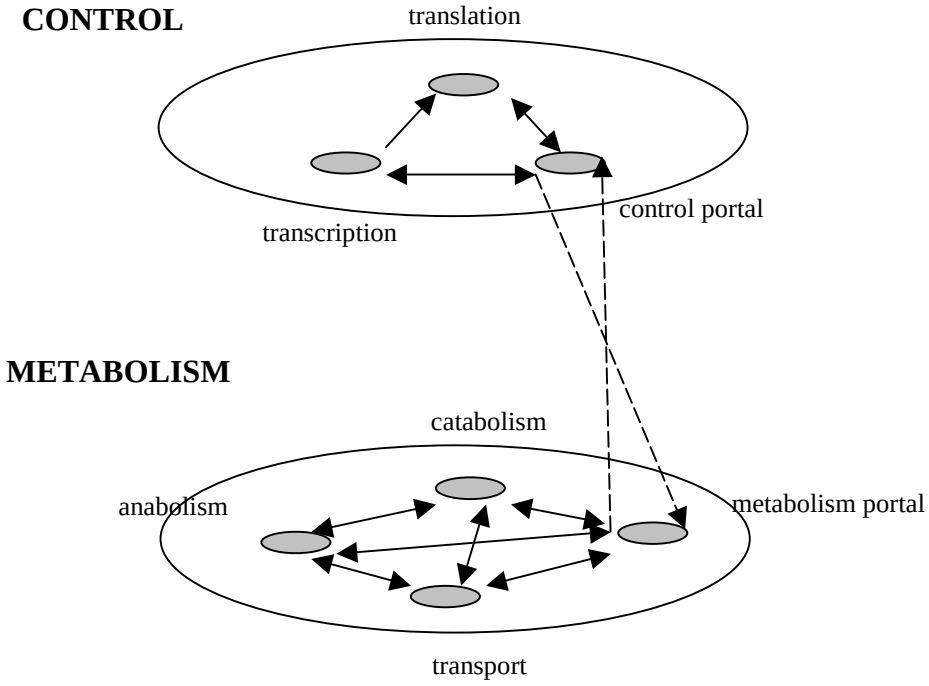


Figure 2. *E.coli* : groups and interactions.

An AGR-model of *E.coli*'s organisational structure is shown in Figure 2. The functional units Control and Metabolism are depicted as different *groups* here (depicted by the larger ovals). Their children (according to Figure 1) are depicted in Figure 2 as *roles* (depicted by smaller ovals) within the groups. The behaviour of these roles, in the next section described by role behaviour properties, is as follows: they receive as input the presence of some substances generated by another role, in order to generate the presence of some new substances as output. The solid arrows represent *intragroup role transfers*, the transfer of substances between roles: they express that a substance produced by one role is used by another role. Notice that each group contains an additional Portal role. The idea is that these roles collect the output substances produced by all other roles within their group, to be able to interact with the other group. The dashed arrows between both portal roles represent *intergroup role interactions*, relating the input of one portal role to the output of the other. Note that the model depicted in Figure 2 is a simplification of the true living cell. For example, only control at the transcriptional and translational level is included, and 'post translational modifications' (such as phosphorylation) are left out. Nevertheless, it reflects the main aspects of its organisational structure in a way that is understandable.

3. Organisation dynamics

The AGR organisation modelling approach was extended with a dynamic modelling approach in (Ferber *et al.*, 2001). To characterise the dynamics within an organisation, dynamic properties of various types can be formulated. For example, a dynamic property of the organisation as a whole, such as

> If oxygen, resources and some nutrients are externally available, then the cell will produce CO_2.

Other examples are dynamic properties of one specific role within an organisation, or dynamic properties that characterise how two roles cooperate.

An organisational structure provides a basis to distinguish in a systematic manner dynamic properties for different elements and aggregation levels within the organisation. In particular, as an extension of the AGR organisation model dynamic properties can be specified for each of the following aggregation levels within the model:

I. At the (highest) aggregation level of the organisation as a whole

- dynamic properties for the *organisation* as a whole; the highest aggregation level, relating any roles within the organisation over time;
- dynamic properties for *intergroup role interaction,* relating the input of one role to the output of a role in another group;

II. At the aggregation level of a group within the organisation

- dynamic properties at the level of a *group*, relating states of roles within a given group over time;
- dynamic properties for *transfer* between roles within a group (from output state of the source role to input state of the destination role);

III. At the (lowest) aggregation level of a role within a group

- dynamic properties at the level of a *role* within a group, relating input and output state (and possibly internal state) of the role;

To describe the dynamics of *E.coli*'s intracellular processes, all types of dynamic properties are used.

3.1. Dynamic Properties of the Organisation as a Whole

The example model for *E. coli*'s dynamics was inspired by the model described in (Jonker, Snoep, Westerhoff and Wijngaards, 2002), which is based on a different modelling approach: the compositional organisation modelling approach. For the example of the living cell, global properties of the organisation as a whole can be expressed in terms of interaction with an Environment. Note that this environment is not shown in Figure 1 and 2, since we consider it not being part of the organisation itself. The cell can use as input from the environment the (external) presence of glucose, gluconate, lactose, O_2, N, P and S. It may export CO_2, ethanol and acetate to the environment. For example, CP1 in Box 1 specifies the property that if O_2 is externally available, as well as resources and at least one of the nutrients glucose, lactose, gluconate, then the cell produces CO_2. Moreover, CP2 specifies an analogue property for the anaerobic case. Note that in addition to d1, w1, also α is a variable, which makes it possible to have different

instantiations of one property. For instance, property CP1(d1, w1, α) may be instantiated to CP1(0.3, 0.5, glucose). For all properties, notice that it is explicitly mentioned when interaction with the environment is involved. More specifically, if by transport a substance is emitted to the environment, this is phrased as 'exports to the Environment', and if a substance is available for transport (i.e., import) within the environment, this is phrased as 'is present within the Environment'. In contrast, the internal exchange of the presence of substances within the organisation model is indicated by the words *generates* and *receives*. For α ranging over {glucose, lactose, gluconate}, the properties shown in Box 1 characterise the cell-environment dynamics.

CP1(d1, w1, α) CO_2 production
At any point in time t
if within the Environment the substances α, O_2, N, P and S are present
then there exists a time point t' with t+d1 ≤ t' ≤ t+w1 such that at t'
the cell exports CO_2 to the Environment

CP2(d2, w2, α) Acetate and ethanol production
At any point in time t
if within the Environment the substances α, N, P and S are present
and within the Environment the substance O_2 is not present
then there exists a time point t' with t+d2 ≤ t' ≤ t+w2 such that at t'
the cell exports acetate and ethanol to the Environment

Box 1. Dynamic properties of the cell as a whole.

Within Computer Science and Artificial Intelligence a number of high-level specification languages have been developed to specify dynamic properties with mathematical precision, thereby allowing qualitative and (sometimes) quantitative aspects. To formally express the properties presented in this paper, the high-level Temporal Trace Language (TTL) has been chosen, introduced in (Jonker and Treur, 1998), to model and analyse the internal and external dynamics of agents, and of multi-agent organisations.

A *trace* or *trajectory* in the state space is a sequence of states indexed over time. *States* are characterised by *state properties* indicating, for example, value assignments to certain variables. *Dynamic properties* are properties of traces, i.e., properties that relate states over time. To express dynamic properties the sorted predicate logic temporal trace language TTL is used. This language is built on atoms referring to, e.g., a *trace* γ, a *time point* t and a *state property* p, such as

'in trace γ at time point t state property p holds'

formalised by

state(γ, t) |== p.

As an example, formalising dynamic property CP1 from Box 1 in TTL yields the following:

$\forall t$ [state(γ, t) |== in_environment(α) $\wedge$
state(γ, t) |== in_environment(O_2) $\wedge$
state(γ, t) |== in_environment(N) $\wedge$
state(γ, t) |== in_environment(P) $\wedge$
state(γ, t) |== in_environment(S) $\Rightarrow$
$\exists t'$ t + d1 $\leq$ t' $\leq$ t + w1 &
state(γ, t') |== cell_exports(CO_2)]

The Temporal Trace Language TTL can play a useful role in modelling complex phenomena from an agent-oriented perspective in the following manners:

- it provides a way to obtain well-defined and mathematically formalisable *specifications* of *dynamic properties* of externally observable agent behaviour, their internal processes, and their organisation; such dynamic properties can be specified at any level of precision as desired.
- for further *analysis* it supports the identification of formalised relationships between different dynamic properties, for example between properties of an agent's externally observable behaviour and its internal processes, or between properties of externally observable agent behaviour and properties of an organisation in which they function.
- it offers possibilities to specify and execute *simulation models* in a high level language, for example simulation of an agent's externally observable behaviour on the basis of its internal processes, or simulation of an organisation on the basis of given or assumed properties of externally observable behaviour of the agents involved.

Throughout the remainder of this paper, dynamic properties will not be formally expressed, but in the semi-formal format presented earlier, to enhance readability. Within this format, each property always holds for all traces γ over the ontology, but γ is not mentioned explicitly to keep the notation simple.

3.2. Intergroup Role Interaction Properties

Within the AGR organisation modelling approach intergroup role interaction properties model connections between groups by specifying how input state of a role in one group can be (temporally) related to output state of another role in a different group. Within the current example, the intergroup role interaction properties take care of the exchange of substances between both groups. This is done by relating the input of the portal role of one group to the output of the portal role of the other group. The properties expressing this are shown in Box 2. The delay parameters in these intergroup role interaction properties can be used to model some form of mobility of molecules produced by one process before they are used in another process. However, for simplicity we assume the exchange to be instantaneous, all delays (ci's and ri's) are 0 in this example, i.e. t' = t in the properties above.

IGIP1(c1, r1) Control Portal-Metabolism Portal Intergroup Role Interaction
At any point in time t, for all substances β,
if Control Portal receives a substance β
then there exists a time point t' with $t+c1 \leq t' \leq t+r1$ such that at t'
Metabolism Portal generates the substance β

IGIP2(c2, r2) Metabolism Portal-Control Portal Intergroup Role Interaction
At any point in time t, for all substances β
if Metabolism Portal receives a substance β
then there exists a time point t' with $t+c2 \leq t' \leq t+r2$ such that at t'
Control Portal generates the substance β

Box 2. Dynamic properties for Intergroup Role Interaction.

3.3. Dynamic Properties of the Metabolism and Control Group

For each of the groups, dynamic properties are considered that contribute to the properties of the organisation as a whole. A group property is specified in terms of temporal relationships between input and output states of roles within this group.

Within the group Metabolism, which includes transportation through the cell's membrane (import and export), substances present outside the cell, but also substances produced by Control can be used. Likewise, it can produce substances that are exported to the environment, as well as substances used by Control. The exchange of substances to and from Control goes via the Metabolism Portal role. Metabolism property MP4 is an example of a complex property that has input from and output to both the environment and Control. For α ranging over {glucose, lactose, gluconate}, the dynamic properties in Box 3 characterise the Metabolism dynamics. As opposed to Metabolism, the group Control does not interact with the environment. Via its role Control Portal certain substances produced by Metabolism are available, and (abstracting from intermediate steps) it can itself produce particular enzymes, ADP, and P. In Box 4 the dynamic properties for the Control group are shown.

3.4. Transfer Properties

Transfer properties are assumed to have a generic pattern: that every transfer generated (in its output state) by any role r1 for any role r2 is received (in its input state) by role r2. In the example, for transfer properties similar assumptions are used as for intergroup role interaction properties, namely instantaneous transfer of all substances (i.e., no time durations taken into account for molecule mobility between chemical processes; all gi's and hi's are 0). All solid arrows in Figure 2 stand for transfer properties. Since they all look the same, only two examples are shown in Box 5. Furthermore, notice that there is a transfer from Transcription to Translation, but not vice versa. For all other combination of roles, there is transfer in two directions.

MP0(m_{init}) Metabolism Initialisation
there exists a time point t with $0 \leq t \leq m_{init}$ such that at t
Metabolism Portal generates the substances ATP, nucleotides and aminoacids

MP1(p1, q1) Metabolism CRPcCAMP production
At any point in time t
if within the Environment the substance glucose is not present
then there exists a time point t' with $t+p1 \leq t' \leq t+q1$ such that at t'
Metabolism Portal receives the substance CRPcAMP

MP2(p2, q2) Metabolism allolactose production
At any point in time t
if within the Environment the substance lactose is present
then there exists a time point t' with $t+p2 \leq t' \leq t+q2$ such that at t'
Metabolism Portal receives the substance allolactose

MP3(p3, q3) Metabolism gluconate_6P production
At any point in time t
if within the Environment the substance gluconate is present
then there exists a time point t' with $t+p3 \leq t' \leq t+q3$ such that at t'
Metabolism Portal receives the substance gluconate_6P_observation_amount

MP4(p4, q4, α)Metabolism ATP-nucleotides-aminoacids production and CO_2 export
At any point in time t
if within the Environment the substances α, N, P, S and O_2 are present
and Metabolism Portal generates the substances ADP, P, respiration_enzymes and import_enzymes for α
then there exists a time point t' with $t+p4 \leq t' \leq t+q4$ such that at t'
Metabolism Portal receives the substances ATP, nucleotides and aminoacids
and the cell exports CO_2 to the Environment

MP5(p5, q5, α)Metabolism ATP-nucleotides-aminoacids production/acetate-ethanol export
At any point in time t
if within the Environment the substances α, N, P and S are present
and within the Environment the substance O_2 is not present
and Metabolism Portal generates the substances ADP, P, fermentation_enzymes and import_enzymes for α
then there exists a time point t' with $t+p5 \leq t' \leq t+q5$ such that at t'
Metabolism Portal receives the substances ATP, nucleotides and aminoacids
and the cell exports acetate and ethanol to the Environment

MP6(p6, q6) Metabolism ArcB_P production
At any point in time t
if within the Environment the substance O_2 is present
then there exists a time point t' with $t+p6 \leq t' \leq t+q6$ such that at t'
Metabolism Portal receives the substance ArcB_P

Box 3. Dynamic properties for the Metabolism group.

CoP1(u1, v1) Glucose_import_enzymes production
At any point in time t
if Control Portal generates the substances nucleotides, ATP and aminoacids
then there exists a time point t' with $t+u1 \leq t' \leq t+v1$ such that at t'
Control Portal receives the substances ADP, P and glucose_import_enzymes

CoP2(u2, v2) Respiration_enzymes production
At any point in time t
if Control Portal generates the substances ArcB_P, nucleotides, ATP and aminoacids
then there exists a time point t' with $t+u2 \leq t' \leq t+v2$ such that at t'
Control Portal receives the substances ADP, P and respiration_enzymes

CoP3(u3, v3) Fermentation_enzymes production
At any point in time t
if Control Portal generates the substances nucleotides, ATP and aminoacids
and Control Portal does not generate the substance ArcB_P
then there exists a time point t' with $t+u3 \leq t' \leq t+v3$ such that at t'
Control Portal receives the substances ADP, P and fermentation_enzymes

CoP4(u4, v4) Lactose_import_enzymes production
At any point in time t
if Control Portal generates the substances allolactose, CRPcAMP, nucleotides, ATP and aminoacids
then there exists a time point t' with $t+u4 \leq t' \leq t+v4$ such that at t'
Control Portal receives the substances ADP, P and lactose_import_enzymes

CoP5(u5, v5) Gluconate_import_enzymes production
At any point in time t
if Control Portal generates the substances gluconate_6P_observation_amount, CRPcAMP, nucleotides, ATP and aminoacids
then there exists a time point t' with $t+u5 \leq t' \leq t+v5$ such that at t'
Control Portal receives the substances ADP, P and gluconate_import_enzymes

Box 4. Dynamic properties for the Control group.

TP1(g1, h1) Transcription-Translation Transfer
At any point in time t, for all substances β
if Transcription generates a substance β
then there exists a time point t' with $t+g1 \leq t' \leq t+h1$ such that at t'
Translation receives the substance β

TP2(g2, h2) Transcription-Control Portal Transfer
At any point in time t, for all substances β
if Transcription generates a substance β
then there exists a time point t' with $t+g2 \leq t' \leq t+h2$ such that at t'
Control Portal receives the substance β

Box 5. Dynamic properties for transfer within the Control group.

3.5. Role Behaviour Properties

Dynamic properties for a role characterise how the role behaves, given its input. Such a dynamic property typically is expressed in terms of a temporal relationship between input state and output state of the role.

Roles within the Control group

The role Transcription can receive the substances nucleotides, ATP, ArcB_P, allolactose, CRPcAMP, and gluconate6P observation amount, all coming (via the Control Portal) from the Metabolism group. Depending on certain circumstances it will produce particular forms of mRNA, ADP, and P. See Box 6 for the dynamic properties of Transcription. The role Translation's input is amino acids and ATP (both produced by the Metabolism group), and a particular type of mRNA (produced by the Transcription role). It can produce ADP, P, and a particular enzyme, corresponding to the type of mRNA. For η ranging over {respiration, fermentation, glucose_import, lactose_import, gluconate_import}, the property in Box 7 characterises the Translation dynamics.

TcP1(k1, l1) Glucose_import_mRNA production
At any point in time t
if Transcription receives the substances nucleotides and ATP
then there exists a time point t' with $t+k1 \leq t' \leq t+l1$ such that at t'
Transcription generates the substances ADP, P and glucose_import_mRNA

TcP2(k2, l2) Respiration_mRNA production
At any point in time t
if Transcription receives the substances ArcB_P, nucleotides and ATP
then there exists a time point t' with $t+k2 \leq t' \leq t+l2$ such that at t'
Transcription generates the substances ADP, P and respiration_mRNA

TcP3(k3, l3) Fermentation_mRNA production
At any point in time t
if Transcription receives the substances nucleotides and ATP
and Transcription does not receive the substance ArcB_P
then there exists a time point t' with $t+k3 \leq t' \leq t+l3$ such that at t'
Transcription generates the substances ADP, P and fermentation_mRNA

TcP4(k4, l4) Lactose_import_mRNA production
At any point in time t
if Transcription receives the substances allolactose, CRPcAMP, nucleotides and ATP
then there exists a time point t' with $t+k4 \leq t' \leq t+l4$ such that at t'
Transcription generates the substances ADP, P and lactose_import_mRNA

TcP5(k5, l5) Gluconate_import_mRNA production
At any point in time t
if Transcription receives the substances gluconate6P_observation_amount, CRPcAMP, nucleotides and ATP
then there exists a time point t' with $t+k5 \leq t' \leq t+l5$ such that at t'
Transcription generates the substances ADP, P and gluconate_import_mRNA

Box 6. Dynamic properties for Transcription.

TlP1(e1, f1, η) Enzymes production
At any point in time t
if Translation receives the substances aminoacids, ATP and mRNA for η
then there exists a time point t' with $t+e1 \leq t' \leq t+f1$ such that at t'
Translation generates the substances ADP, P and enzymes for η

Box 7. Dynamic properties for Translation.

Roles within the Metabolism group

Within the Metabolism group, the role Catabolism receives the presence of several substances. Some of them are provided by the other roles, Anabolism and Transport, some others are provided (via the Metabolism Portal) by the Control group. Likewise, the substances it produces are also used by the roles Anabolism and Transport, and (via the Metabolism Portal) by the group Control. For δ ranging over {glucose6P, gluconate6P, lactose}, the dynamic properties shown in Box 8 characterise the Catabolism dynamics.

CaP0(c_{init}) Catabolism Initialisation
there exists a time point t with $0 \leq t \leq c_{init}$ such that at t
Catabolism generates the substance NAD(P)

CaP1(i1, j1, δ) Catabolism Dynamics 1
At any point in time t
if Catabolism receives substances δ, ADP, P, NAD(P), O_2 and respiration_enzymes
then there exists a time point t' with $t+i1 \leq t' \leq t+j1$ such that at t'
Catabolism generates the substances glucose6P, pyruvate, ATP, NAD(P)H, PEP and CO_2

CaP2(i2, j2, δ) Catabolism Dynamics 2
At any point in time t
if Catabolism receives the substances δ, ADP, P, NAD(P) and fermentation_enzymes
then there exists a time point t' with $t+i2 \leq t' \leq t+j2$ such that at t'
Catabolism generates the substances glucose6P, pyruvate, ATP, NAD(P)H, PEP, acetate and ethanol

Box 8. Dynamic properties for Catabolism.

AP0(a_{init}) Anabolism Initialisation
there exists a time point t with $0 \leq t \leq a_{init}$ such that at t
Anabolism generates the substances PEP and ATP

AP1(m1, n1) Anabolism Dynamics 1
At any point in time t
if Anabolism receives substances ATP, NAD(P)H, glucose6P, pyruvate, N, P and S
then there exists a time point t' with $t+m1 \leq t' \leq t+n1$ such that at t'
Anabolism generates the substances ADP, P, NAD(P), nucleotides and aminoacids

Box 9. Dynamic properties for Anabolism.

TpP1(s1, t1) Transport CRPcAMP production
At any point in time t
if within the Environment the substance glucose is not present
then there exists a time point t' with t+s1 ≤ t' ≤ t+t1 such that at t'
Transport generates the substance CRPcAMP

TpP2(s2, t2) Transport allolactose production
At any point in time t
if within the Environment the substance lactose is present
then there exists a time point t' with t+s2 ≤ t' ≤ t+t2 such that at t'
Transport generates the substance allolactose

TpP3(s3, t3) Transport gluconate_6P_observation_amount production
At any point in time t
if within the Environment the substance gluconate is present
then there exists a time point t' with t+s3 ≤ t' ≤ t+t3 such that at t'
Transport generates the substance gluconate6P_observation_amount

TpP4(s4, t4) Transport glucose6P production
At any point in time t
if Transport receives the substances PEP and glucose_import_enzymes
and within the Environment the substance glucose is present
then there exists a time point t' with t+s4 ≤ t' ≤ t+t4 such that at t'
Transport generates the substances glucose6P and pyruvate

TpP5(s5, t5) Transport gluconate6P production
At any point in time t
if Transport receives the substances ATP and gluconate_import_enzymes
and within the Environment the substance gluconate is present
then there exists a time point t' with t+s5 ≤ t' ≤ t+t5 such that at t'
Transport generates the substances gluconate6P, ADP and P

TpP6(s6, t6) Transport lactose production
At any point in time t
if Transport receives the substances ATP and lactose_import_enzymes
and within the Environment the substance lactose is present
then there exists a time point t' with t+s6 ≤ t' ≤ t+t6 such that at t'
Transport generates the substances lactose, ADP and P

TpP7(s7, t7) Transport O_2 production
At any point in time t
if within the Environment the substance O_2 is present
then there exists a time point t' with t+s7 ≤ t' ≤ t+t7 such that at t'
Transport generates the substances O_2 and ArcB_P

TpP8(s8, t8, ε) Transport resources production
At any point in time t
if Transport receives the substance ATP
and within the Environment the substance ε is present
then there exists a time point t' with t+s8 ≤ t' ≤ t+t8 such that at t'
Transport generates the substances ε, ADP and P

TpP9(s9, t9, ζ) Transport environment export
At any point in time t
if Transport receives the substance ζ
then there exists a time point t' with t+s9 ≤ t' ≤ t+t9 such that at t'
the cell exports ζ to the Environment

Box 10. Dynamic properties for Transport.

Like Catabolism, the role Anabolism also interacts with several roles in its own group. Apart from initialisation, its dynamics can be described by one single property; see Box 9. The role Transport is the only role within the organisation that interacts with the environment. Furthermore, it also interacts with roles in its own group, including the Metabolism Portal role, which serves as a portal to the Control group. Recall that the words *generates* and *receives* indicate that the substances are exchanged within the organisation model *internally*. In contrast, if by transport a substance is emitted to the environment, this is phrased as 'exports to the Environment', and if a substance is available for transport (i.e., import) within the environment, this is phrased as 'is present within the Environment'. For ε ranging over {N, P, S} and ζ ranging over {acetate, ethanol, CO2}, the properties shown in Box 10 characterise the Transport dynamics.

4. Interlevel Relations

The idea of expressing dynamic properties at different levels of aggregation is that certain logical interlevel relationships can be identified between properties at the different levels. Typically, dynamics of the whole organised (multi-agent) system can be related to dynamic group properties and intergroup interaction properties via the following pattern:

dynamic properties for the groups &
dynamic properties for intergroup role interaction ⇒
dynamic properties for the organisation

This implication should be understood as follows: 'for any organisation, if for any trace the group properties and intergroup role interaction properties hold, then the general properties for the organisation also hold'. Likewise, dynamic properties of groups can be related to dynamic properties of roles in the following way:

dynamic properties for roles &
dynamic properties for transfer between roles ⇒
dynamic properties for a group

A general overview of the interlevel relationships between dynamic properties at different aggregation levels is depicted as an AND-tree in Figure 3.

The next sections will describe the interlevel relationships between dynamic properties within the example of the living cell.

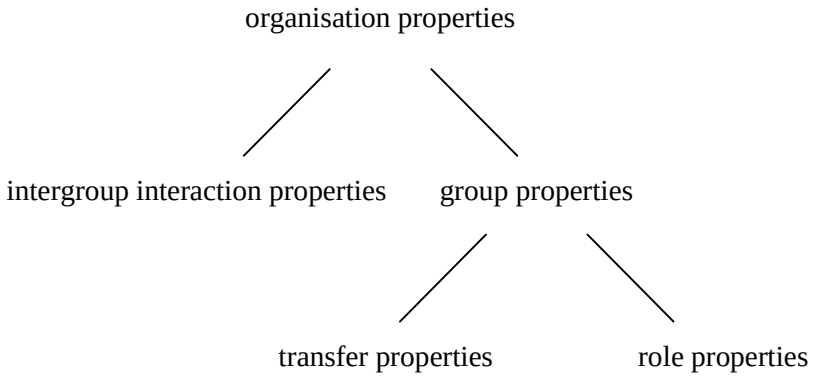


Figure 3. Overview of interlevel relationships between dynamic properties within an organisation model.

4.1. Interlevel Relations for Overall Properties of the Cell Dynamics

Global property CP1(glucose) states that the cell will produce CO_2 if the substances O_2, glucose, N, P and S are available within the environment. Careful investigation of the group properties and intergroup role interaction properties yield the interlevel relationship depicted in Figure 4. The interlevel relationship between Global Property CP1(lactose) and the properties it depends on is depicted in Figure 5. This property states that the cell will produce CO_2 if the substances O_2, lactose, N, P and S are available within the environment. However, nothing is said about the availability of glucose. An argumentation of the dependencies shown could therefore be obtained by reasoning by cases: suppose all lower level properties of Figure 5 hold. Then, if glucose is present within the environment, this will be used in order to export CO_2, according to properties MP0, MP6, IGIP2, CoP1, CoP2, IGIP1, and MP4(glucose). But if glucose is not present and lactose is present within the environment, then lactose will be used, according to properties MP0, MP1, MP2, MP6, IGIP2, CoP2, CoP4, IGIP1, and MP4(lactose). Hence, if all lower level properties hold, then CO_2 will always be exported, making use of either glucose or lactose from the environment. It may thus be concluded that CP1(lactose) holds.

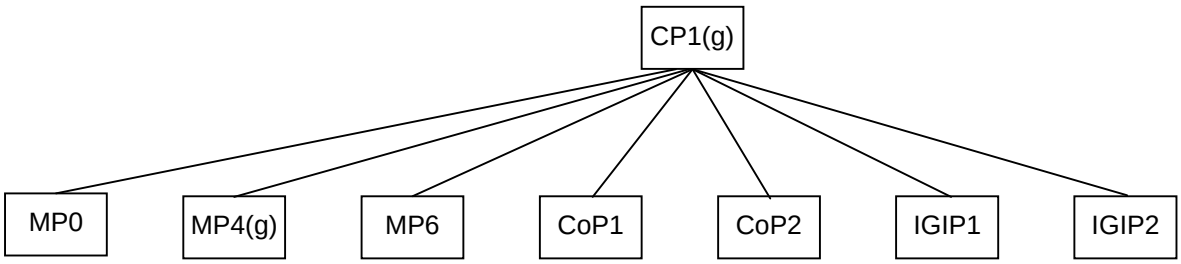


Figure 4. Property CP1(g) related to group properties and intergroup role interaction properties.

A complete specification of the interlevel relations for the global properties is given in Box 11. In addition to the relationships between the properties, dependencies between corresponding parameters are given. Recall that in this example the delays of the intergroup role interaction properties (all c_i's and r_i's) are assumed to be 0, so they could have been left out as well.

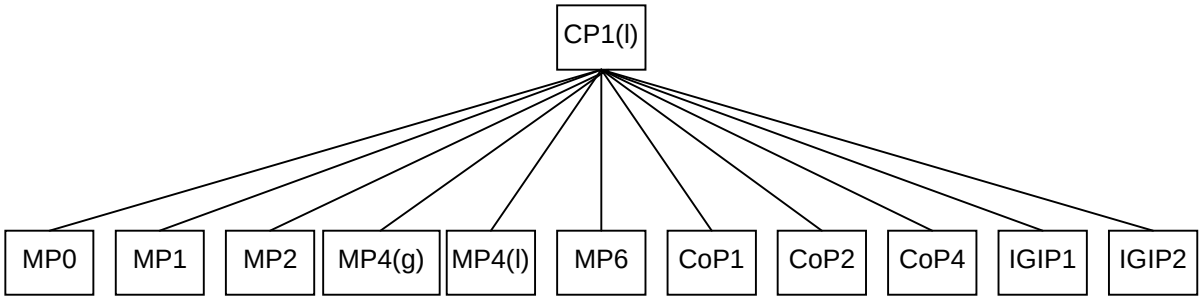


Figure 5. Property CP1(l) related to group properties and intergroup role interaction properties.

MP0(m_{init}) & MP4(p4, q4, glucose) & MP6(p6, q6)
& CoP1(u1, v1) & CoP2(u2, v2)
& IGIP1(c1, r1) & IGIP2(c2, r2) ⇒ CP1(d1, w1, glucose)

with d1 = max(0+c2+u1+c1, max(0+c2, p6+c2)+u2+c1)+p4,
w1 = max(m_{init}+r2+v1+r1, max(m_{init}+r2, q6+r2)+v2+r1)+q4.

MP0(m_{init}) & MP1(p1, q1) & MP2(p2, q2)
& MP4(p4, q4, glucose) & MP4(p4, q4, lactose) & MP6(p6, q6)
& CoP1(u1, v1) & CoP2(u2, v2) & CoP4(u4, v4)
& IGIP1(c1, r1) & IGIP2(c2, r2) ⇒ CP1(d1, w1, lactose)

with d1glucose = max(0+c2+u1+c1, max(0+c2, p6+c2)+u2+c1)+p4,
w1glucose = max(m_{init}+r2+v1+r1, max(m_{init}+r2, q6+r2)+v2+r1)+q4,
d1lactose = max(max(0+c2, p6+c2)+u2+c1, max(0+c2, p1+c2, p2+c2)+u4+c1)+p4,
w1lactose = max(max(m_{init}+r2, q6+r2)+v2+r1, max(m_{init}+r2, q1+r2, q2+r2)+v4+r1)+q4,
d1 = min(d1glucose, d1lactose),
w1 = max(w1glucose, w1lactose).

MP0(m_{init}) & MP1(p1, q1) & MP3(p3, q3)
& MP4(p4, q4, glucose) & MP4(p4, q4, gluconate) & MP6(p6, q6)
& CoP1(u1, v1) & CoP2(u2, v2) & CoP5(u5, v5)
& IGIP1(c1, r1) & IGIP2(c2, r2) ⇒ CP1(d1, w1, gluconate)

with d1glucose = max(0+c2+u1+c1, max(0+c2, p6+c2)+u2+c1)+p4,
w1glucose = max(m_{init}+r2+v1+r1, max(m_{init}+r2, q6+r2)+v2+r1)+q4,
d1gluconate = max(max(0+c2, p6+c2)+u2+c1, max(0+c2, p1+c2, p3+c2)+u5+c1)+p4,
w1gluconate = max(max(m_{init}+r2, q6+r2)+v2+r1,
max(m_{init}+r2, q1+r2, q3+r2)+v5+r1)+q4,
d1 = min(d1glucose, d1gluconate),
w1 = max(w1glucose, w1gluconate).

MP0(m_{init}) & MP5(p5, q5, glucose)
& CoP1(u1, v1) & CoP3(u3, v3)
& IGIP1(c1, r1) & IGIP2(c2, r2) ⇒ CP2(d2, w2, glucose)

with d2 = 0+c2+max(u1+c1, u3+c1)+p5,
w2 = m_{init}+r2+max(v1+r1, v3+r1)+q5**.**

MP0(m_{init}) & MP1(p1, q1) & MP2(p2, q2)
& MP5(p5, q5, glucose) & MP5(p5, q5, lactose)
& CoP1(u1, v1) & CoP3(u3, v3) & CoP4(u4, v4)
& IGIP1(c1, r1) & IGIP2(c2, r2) ⇒ CP2(d2, w2, lactose)

with d2glucose = 0+c2+max(u1+c1, u3+c1)+p5,
w2glucose = m_{init}+r2+max(v1+r1, v3+r1)+q5,
d2lactose = max(0+c2+u3+c1, max(0+c2, p1+c2, p2+c2)+u4+c1)+p5,
w2lactose = max(m_{init}+r2+v3+r1, max(m_{init}+r2, q1+r2, q2+r2)+v4+r1)+q5,
d2 = min(d2glucose, d2lactose),
w2 = max(w2glucose, w2lactose).

MP0(m_{init}) & MP1(p1, q1) & MP3(p3, q3)
& MP5(p5, q5, glucose) & MP5(p5, q5, gluconate)
& CoP1(u1, v1) & CoP3(u3, v3) & CoP5(u5, v5)
& IGIP1(c1, r1) & IGIP2(c2, r2) ⇒ CP2(d2, w2, gluconate)

with d2glucose = 0+c2+max(u1+c1, u3+c1)+p5,
w2glucose = m_{init}+r2+max(v1+r1, v3+r1)+q5,
d2gluconate = max(0+c2+u3+c1, max(0+c2, p1+c2, p3+c2)+u5+c1)+p5,
w2gluconate = max(m_{init}+r2+v3+r1, max(m_{init}+r2, q1+r2, q3+r2)+v5+r1)+q5,
d2 = min(d2glucose, d2gluconate),
w2 = max(w2glucose, w2gluconate).

Box 11. Interlevel relations for the dynamic properties of the cell as a whole.

To explain the idea of the interlevel relationships, and in particular, the relations between the time durations involved, in some more detail, Figure 6 depicts a Flow Diagram for property CP1(glucose). Nodes represent the presence of certain substances in certain places at a certain time. (Combinations of) edges represent properties. As can be seen in the picture, the cell is able to export CO_2 to the environment if the properties MP0, MP4(glucose), MP6, CoP1, CoP2, IGIP1 and IGIP2 hold. Figure 6 can be useful for understanding of the dependencies between the parameter values given above. Namely, when two processes occur in a sequence, the time duration of this sequence equals the sum of both individual time durations. For instance, the minimal duration assigned to the sequence of processes described by properties MP0(0, m_{init})-IGIP2(c2, r2) is 0+c2, the maximal duration is m_{init}+r2. Likewise, the time duration of the combination of two processes occurring in parallel equals the maximum of the individual time durations. In this case the processes have to be synchronised. Thus, if process C needs the simultaneous output of the parallel processes A and B as input, C can only start when both A and B have finished, under the assumption that the output substances of the processes persist long enough in order to co-occur at the same time instance.

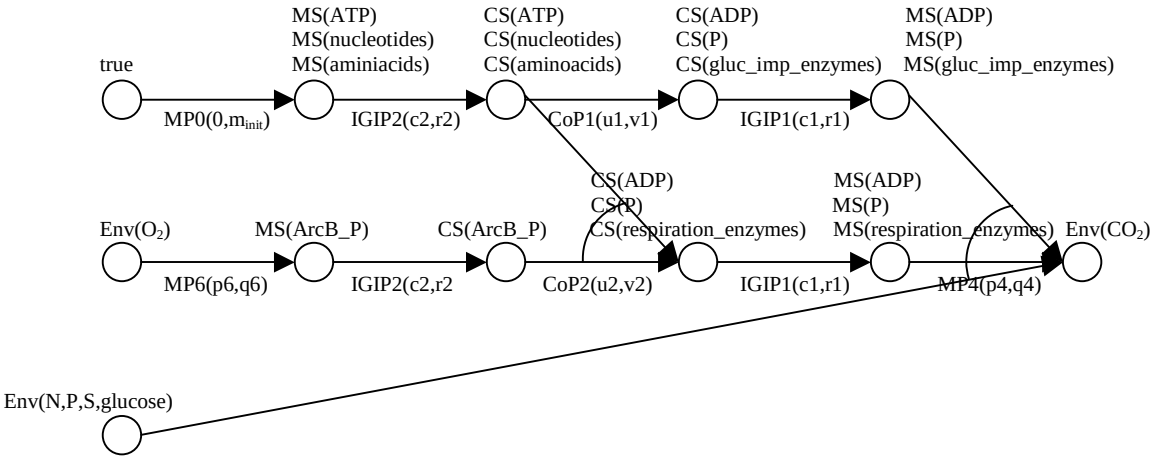


Figure 6. Flow Diagram for Property CP1(glucose).

4.2. Interlevel Relations for Group Properties: Metabolism Dynamics

As shown in Figure 7, Metabolism group property MP6 is related to role behaviour property TpP7, together with the transfer properties of Metabolism, indicated by TP(M). A complete specification of the interlevel relations for all group properties of Metabolism is given in Box 12. Since all transfers are assumed to be instantaneous, the parameters of TP(M) have no influence.

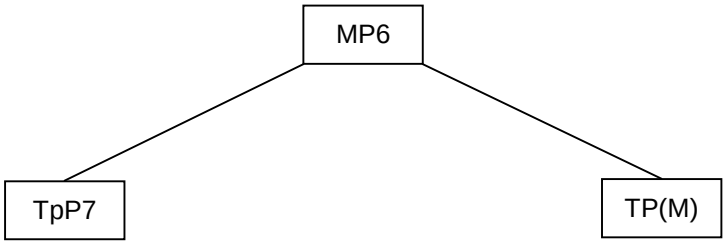


Figure 7. Property MP6 related to role behaviour property and transfer properties.

TpP1(s1, t1) & TP(M) ⇒ MP1(p1, q1)

with p1 = s1, q1 = t1**.**

TpP2(s2, t2) & TP(M) ⇒ MP2(p2, q2)

with p2 = s2, q2 = t2**.**

TpP3(s3, t3) & TP(M) ⇒ MP3(p3, q3)

with p3 = s3, q3 = t3**.**

TpP4(s4, t4) & TpP7(s7, t7) & TpP8(s8, t8, N) & TpP8(s8, t8, P) & TpP8(s8, t8, S) & TpP9(s9, t9, CO_2) & CaP0(c_{init}) & CaP1(i1, j1, glucose6P) & AP0(a_{init}) & AP1(m1, n1) & TP(M) ⇒ MP4(p4, q4, glucose)

with p4 = max(0, s7, 0+s8, 0+s4)+i1+max(s9, m1),
q4 = max(c_{init}, t7, a_{init}+t8, a_{init}+t4)+j1+max(t9, n1).

TpP6(s6, t6) & TpP7(s7, t7) & TpP8(s8, t8, N) & TpP8(s8, t8, P) & TpP8(s8, t8, S) & TpP9(s9, t9, CO_2) & CaP0(c_{init}) & CaP1(i1, j1, lactose) & AP0(a_{init}) & AP1(m1, n1) & TP(M) ⇒ MP4(p4, q4, lactose)

with p4 = max(0, s7, 0+s8, 0+s6)+i1+max(s9, m1),
q4 = max(c_{init}, t7, a_{init}+t8, a_{init}+t6)+j1+max(t9, n1).

TpP5(s5, t5) & TpP7(s7, t7) & TpP8(s8, t8, N) & TpP8(s8, t8, P) & TpP8(s8, t8, S) & TpP9(s9, t9, CO_2) & CaP0(c_{init}) & CaP1(i1, j1, gluconate6P) & AP0(a_{init}) & AP1(m1, n1) & TP(M) ⇒ MP4(p4, q4, gluconate)

with p4 = max(0, s7, 0+s8, 0+s5)+i1+max(s9, m1),
q4 = max(c_{init}, t7, a_{init}+t8, a_{init}+t5)+j1+max(t9, n1).

TpP4(s4, t4) & TpP8(s8, t8, N) & TpP8(s8, t8, P) & TpP8(s8, t8, S) & TpP9(s9, t9, acetate) & TpP9(s9, t9, ethanol) & CaP0(c_{init}) & CaP2(i2, j2, glucose6P) & AP0(a_{init}) & AP1(m1, n1) & TP(M) ⇒ MP5(p5, q5, glucose)

with p4 = max(0, 0+s8, 0+s4)+i2+max(s9, m1),
q4 = max(c_{init}, a_{init}+t8, a_{init}+t4)+j2+max(t9, n1).

TpP6(s6, t6) & TpP8(s8, t8, N) & TpP8(s8, t8, P) & TpP8(s8, t8, S) & TpP9(s9, t9, acetate) & TpP9(s9, t9, ethanol) & CaP0(c_{init}) & CaP2(i2, j2, lactose) & AP0(a_{init}) & AP1(m1, n1) & TP(M) ⇒ MP5(p5, q5, lactose)

with p4 = max(0, 0+s8, 0+s6)+i2+max(s9, m1),
q4 = max(c_{init}, a_{init}+t8, a_{init}+t6)+j2+max(t9, n1).

TpP5(s5, t5) & TpP8(s8, t8, N) & TpP8(s8, t8, P) & TpP8(s8, t8, S) & TpP9(s9, t9, acetate) & TpP9(s9, t9, ethanol) & CaP0(c_{init}) & CaP2(i2, j2, gluconate6P) & AP0(a_{init}) & AP1(m1, n1) & TP(M) ⇒ MP5(p5, q5, gluconate)

with p4 = max(0, 0+s8, 0+s5)+i2+max(s9, m1),
q4 = max(c_{init}, a_{init}+t8, a_{init}+t5)+j2+max(t9, n1).

TpP7(s7, t7) & TP(M) ⇒ MP6(p6, q6)

with p6 = s7, q6 = t7.

Box 12. Interlevel relations for the dynamic properties of the Metabolism group.

4.3. Interlevel Relations for Group Properties: Control Dynamics

Figure 8 shows how Control group property CoP1 is related to role behaviour properties TlP1(glucose_import) and TcP1, together with all transfer properties of Control, indicated by TP(Co). A complete specification of all interlevel relations for the group properties of Control is given in Box 13.

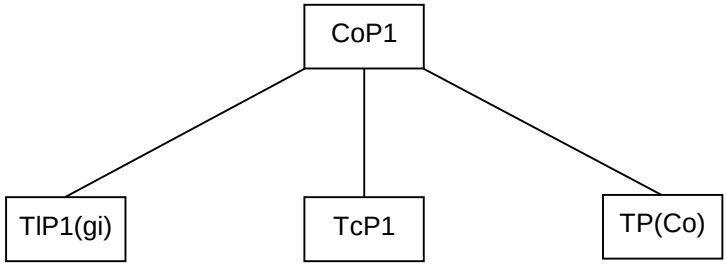


Figure 8. Property CoP1 related to role behaviour properties and transfer properties.

4.4. Diagnosis based on Interlevel Relationships

The dynamic properties as presented above can be formalised in a mathematical-logical manner. Based on such a formalisation, a software environment can and actually has been developed to *automatically check* whether such properties hold for a given (empirical or simulated) trace over time for the dynamics of an organisation.

If the interlevel relationships between the dynamic properties are known, for example as depicted in Figure 4, they can be used for *diagnosis of dysfunctioning* within an organisation. For example, suppose for a given trace at some point in time it has been detected that the dynamic property CP1(glucose) at the highest aggregation level of the organisation does not hold, i.e., the cell does not produce CO_2 although the substances O_2, glucose, N, P and S are available within the environment. Given the AND-tree structure in Figure 4, at least one of the children will not hold (if they all would hold for the given trace, also CP1(glucose) would hold for this trace), which means that either MP0, MP6, IGIP2, CoP1, CoP2, IGIP1, or MP4(glucose) will not hold. Suppose by further checking it is found that MP6 does not hold. Then the diagnostic process can be continued by focusing on this property. It follows that either TpP7 or TP(M) does not hold (see Figure 7). Checking these two properties will pinpoint the cause of the organisation's dysfunctioning. Notice that this diagnostic process is economic in the sense that the whole subtree under e.g. CoP1 is not examined since there is no reason for that, as CoP1 holds.

TlP1(e1, f1, glucose_import) & TcP1(k1, l1) & TP(Co) ⇒ CoP1(u1, v1)
with u1 = k1+e1, v1 = l1+f1**.**

TlP2(e2, f2, respiration) & TcP2(k2, l2) & TP(Co) ⇒ CoP2(u2, v2)
with u2 = k2+e2, v2 = l2+f2**.**

TlP3(e3, f3, fermentation) & TcP3(k3, l2) & TP(Co) ⇒ CoP3(u3, v3)
with u3 = k3+e3, v3 = l3+f3**.**

TlP4(e4, f4, lactose_import) & TcP4(k4, l4) & TP(Co) ⇒ CoP4(u4, v4)
with u4 = k4+e4, v4 = l4+f4**.**

TlP5(e5, f5, gluconate_import) & TcP5(k5, l5) & TP(Co) ⇒ CoP5(u5, v5)
with u5 = k5+e5, v5 = l5+f5**.**

Box 13. Interlevel relations for the dynamic properties of the Control group.

5. Simulation and Checking

A software environment has been created to enable the simulation of executable organisation models specified at a high conceptual level. The input of this simulation environment is a set of dynamic properties. In Section 3.1 the language TTL was introduced as an expressive language for the purpose of specification and checking of dynamic properties. For the purpose of simulation, to obtain computational efficiency the format used for dynamic properties is more restricted than the TTL format used to specify various types of dynamic properties: they are in so-called 'leads to' format. This is a real time-valued variant of Executable Temporal Logic (Barringer *et al.*, 1996). Roughly spoken, in *leads to* format the following can be expressed:

if a certain state property α holds for a certain time interval with duration g, then after some delay (between e and f) another state property β will hold for a certain time interval h

This specific temporal relationship *leads to* is applicable forward as well as backward in time. Hence, if α and β are state properties, and α leads to β, this also means that if β holds for a time interval of length h, then α held during some time interval with length g, of which the starting point was between e and f before the starting point of the second interval.

Making use of these *leads to* properties, the software environment generates simulation traces. A trace is developed by starting at time t = 0 and for each time point up to which the trace already has been constructed, checking which antecedents of executable properties hold in the already constructed trace. For these executable properties, add the consequent to the trace, i.e., extend the trace in time in such a manner that the consequent holds.

The relation between the specification and the constructed trace is that the trace is a model (in the logical sense) of the theory defined by the specification, i.e., all executable dynamic *leads to* properties of the specification hold in the trace.

5.1. Simulation

The software environment described above has been used to simulate the internal dynamics of the organisation of the cell. In order to do this, all lowest level properties

have been expressed in *leads to* format. For this example, these were all intergroup role interaction properties, role behaviour properties and transfer properties. The specific timing parameter values assigned to the role behaviour properties, inspired by (Jonker, Snoep, Westerhoff and Wijngaards, 2002), are given in Table 1. For the other properties, all time parameters were 0.

Property	**Minimal delay (e)**	**Maximal delay (f)**	**Duration antecedent (g)**	**Duration consequent (h)**
CaP1	4	12	4	80
CaP2	4	12	4	4
AP1	2	6	4	4
TpP1	-4	0	4	4
TpP2	0	0	0.23	0.23
TpP3	0	0	0.23	0.23
TpP4	-4	0	4	80
TpP5	-4	0	4	4
TpP6	-4	0	4	4
TpP7	0	0	4	4
TpP8	0	0	4	4
TpP9	0	0	4	4
TcP1	60	60	1	40
TcP2	60	60	1	40
TcP3	60	60	1	40
TcP4	60	60	1	40
TcP5	60	60	1	40
TlP1	0	0	10	600

Table 1. Time parameters for the leads to properties.

In order to initialise the simulation, the truth values of all state properties have been set to *true* from time point 0 to 60. Furthermore, for each simulation run particular settings had to be assigned to the environment. An example situation, where lactose and resources are always present, the presence of glucose and O_2 is fluctuating, and gluconate is always absent, can be seen in Figure 9. In this trace, time is on the horizontal axis, the properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false during that time period.

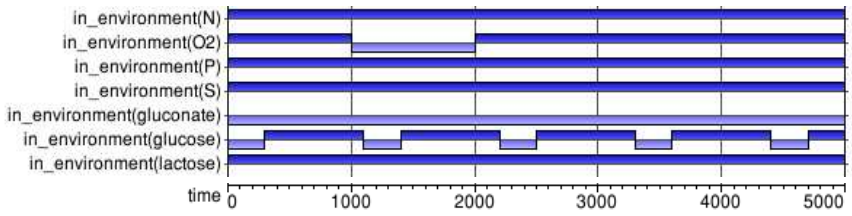


Figure 9. Environmental dynamics.

Another part of this trace, depicting the reaction of the cell to this environment, is shown in Figure 10. Notice that the cell exports acetate, ethanol and CO_2 at the very beginning, because of the initialisation conditions. However, as it adapts to the environment only CO_2 is exported. As the environmental oxygen disappears, the cell's CO_2 emissions stop very soon, and acetate and ethanol are produced instead. After the oxygen re-appears in the environment, the cell adapts by stopping the acetate and ethanol emissions after a while and returning to CO_2 production. Note that the acetate and ethanol emissions are not stopped immediately. This is because the internal substances needed for these emissions (including fermentation enzymes) persist for some time.

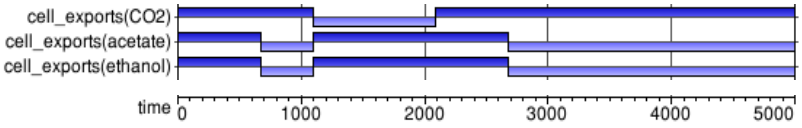


Figure 10. Simulated overall behaviour.

An interesting observation is the fact that the fluctuating presence of glucose in the environment does not seem to have any influence on the production of CO_2, acetate and ethanol. According to the highest level properties CP1 and CP2, this is indeed the correct behaviour, since for the behaviour at this level it does not matter whether it is glucose, lactose, or gluconate, as long as one of the nutrients is available. And in this particular case, lactose is always present in the environment. Nevertheless, the fluctuating presence of glucose does influence the behaviour of the cell at a lower level. For instance, consider the next part of the same trace, depicting the output of the roles Anabolism, Catabolism, Transport, Transcription and Translation, see Figure 11.

Figure 11 shows that the presence of glucose in the environment influences, for instance, the internal production of the substance CRPcAMP by the Transport role. As a consequence, the presence of (among others) this CRPcAMP leads to the creation of lactose_import_mRNA by the Transcription role, whilst glucose_import_mRNA is created continuously. To go one step further, lactose_import_mRNA and glucose_import_mRNA are used by the Translation role to create, with a certain delay, lactose_import_enzymes and glucose_import_enzymes. It can thus be concluded that from an external perspective there is no visible difference in behaviour of the cell, whether there is only lactose outside or both lactose and glucose. Nevertheless, from an internal perspective many differences can be seen. The entire trace resulting from this simulation covers 245 state properties, representing not only the output but also the input state properties of the roles shown above. However, since the transfer of substances is instantaneous and without delay in our model, each output state property for one role results in several identical input state properties for the other roles. Likewise, the input and output state properties of the Metabolism Portal and Control Portal group are identical to state properties already shown above. Hence, for reasons of presentation, the rest of the trace is not shown in this paper.

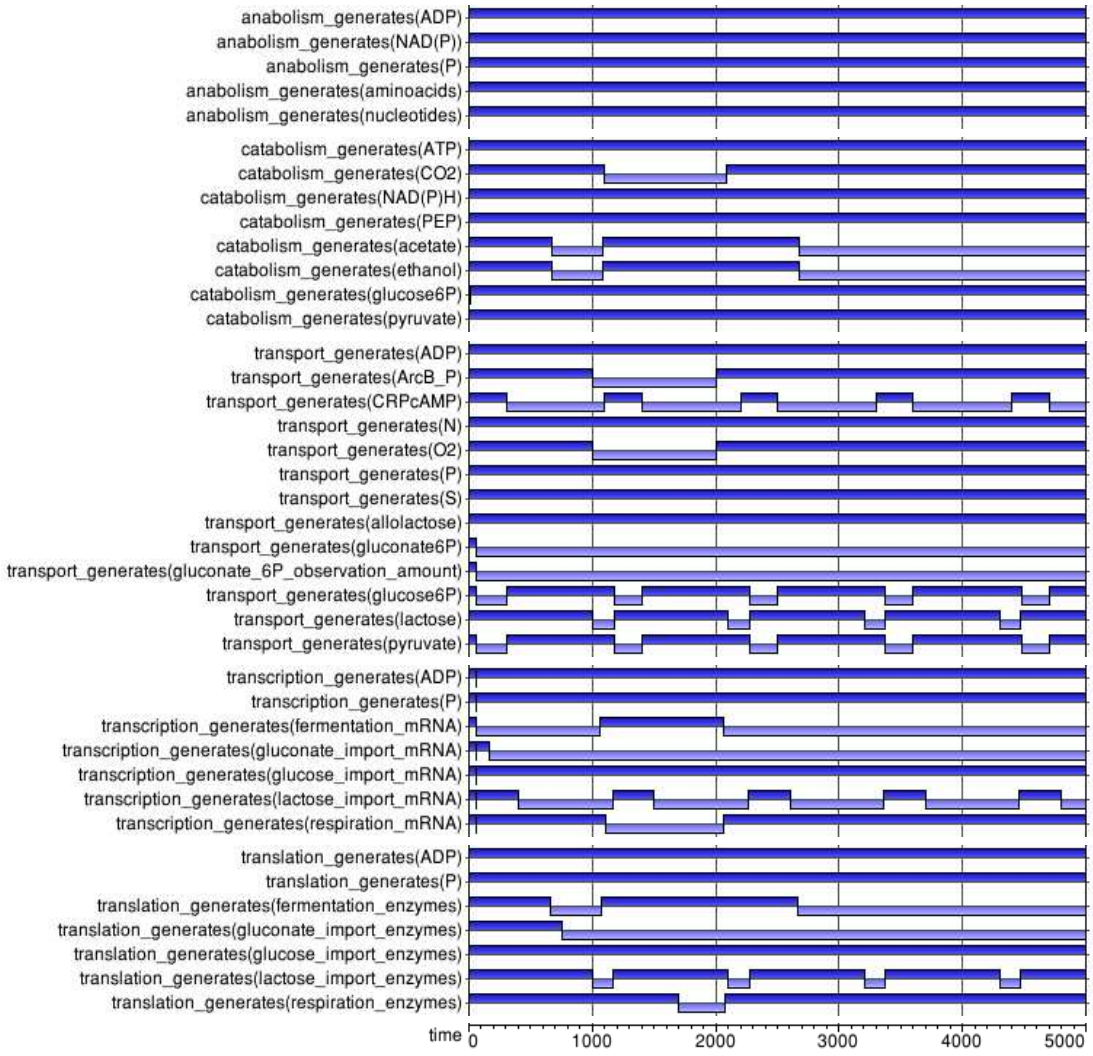


Figure 11. Simulated internal dynamics.

5.2. Checking Properties

As mentioned in Section 4.4, interlevel relationships between properties, as depicted in the tree of Figure 4, can be very useful in the analysis of dynamic properties of an organisation. In order to perform such an analysis, some mechanism is needed to check if a certain property holds for a given trace. To this end, the simulation software described above automatically produces log files containing formal representations of the traces. In addition, software has been developed that is able to read in these formally represented traces together with a set of dynamic properties and to perform the checking process. As a result, the software determines not only whether the properties hold for the trace or not, but in case of failure, it also pinpoints which parts of the trace violate the properties. For our simulation, checks of this kind have actually been performed for all Global Properties and Group Properties, i.e. all properties of Section 3.1 and 3.3. They all turned out to hold for the generated traces. This validates the interlevel relationships.

6. Discussion

Analysis and simulation of biological (and in particular, cellular) processes is a huge research area in which many groups are working, e.g. (Takahashi, 2004). As a novel contribution to this area, the current paper shows how an organisation modelling approach can be used to analyse and simulate the dynamics of biological organisation, illustrated for the functioning of intracellular processes. This biological system can be modeled as consisting of a number of active components or agents that are connected and grouped together in such a manner that everything functions well. Dynamic properties at different levels of aggregation of the organisation model have been identified, and relationships between these dynamic properties at different aggregation levels were made explicit. Based on the executable properties, simulation has been performed and (higher-level) properties have been checked for the produced simulation traces. Thus the interlevel relationships between properties at different aggregation levels have been verified. This case study shows that organisation modelling techniques can play a useful role in biological application areas.

6.1. The organisation modelling perspective

The analysis method for the dynamics from an organisation modelling perspective involves the following ingredients:

- Specify state properties and dynamic properties of the overall process
- Identify the agents and their roles within the overall process
- Specify state properties and dynamic properties for the behaviour of these roles
- Identify groups of roles
- Specify dynamic properties for groups
- Specify dynamic intergroup role interaction and transfer properties
- Identify interlevel relations between dynamic properties at different levels of aggregation: relating role, group and organisation dynamics
- Relate state properties to physical or chemical state properties
- Relate dynamic properties to physical or chemical dynamic properties
- Specify executable dynamic properties
- Simulate dynamics based on executable dynamic properties
- Check given traces of dynamics against dynamic properties

Software support for some of these items within analysis has been developed or is under development. For example, an editor to specify dynamic properties according to a specific format, a theorem prover that relates (specific types of) dynamic properties to other dynamic properties, and a (model) checker that checks whether dynamic properties hold in a given trace; e.g., (Jonker, Letia, and Treur, 2002; Jonker, Treur, and Wijngaards, 2002).

6.2. Organisation as a unifying perspective on addressing and developing biocomplexity

In another case study the organisation of the circulatory system in mammals was addressed; cf. (Bosse, Jonker and Treur, 2004). The two biological case studies, the circulatory system and the living cell, have some aspects in common and differ in some other aspects. A main common aspect is that in both cases Nature shows a certain form of organisation. Although the areas are quite distinct, the organisation modelling approach illustrates this common aspect by using generic concepts such as roles and groups to model both example processes. This is a main contribution of this paper: to show that the notion of organisation as observed in practically all areas in Nature, can be addressed and formalised from a generic perspective. Thus a unifying perspective is obtained on the way how Nature copes with (and develops) complexity by exploiting (increasing degrees of) organisation. Organisation modelling techniques as put forward in this paper provide means to describe, compare and distinguish the different forms and principles of organisation possible and/or occurring in Nature, thus providing a structuring of the variety of biodiversity and biocomplexity from the perspective of underlying organisational principles.

6.3. Hard-wired versus emerging organisational structure

The two examples described in the previous section - circulatory system and intracellular processes - also illustrate differences. In the circulatory system modelling the organisation structure is in some sense ‘hard-wired’ in physical reality. Arteries and veins are physically connected to heart, lungs and other organs, and each of the organs has a specific location, which is non-overlapping with locations of other organs. The functioning of this organisation is forced by this physical configuration. For example, if an artery is cut off or a vein is decoupled from the heart, then the entailed dysfunctioning of the organisation usually is lethal for the organism. In contrast to this ‘hard-wired’ case, the living cell example shows a kind of opposite situation. Here all processes are assumed to occupy the same spatial area. No fixed physical separations and connections between the various processes exist (as would be the case in an installation in a chemical factory), except that all substances are kept together within the cell by the membrane (the soup metaphor). Escape is only possible in some cases, which are often controlled by the cell. Within the cell free mobility is assumed for all substances involved. In this case the functioning of the organisation emerges from the possibilities for the ways in which the various chemical processes can interact with each other.

One might expect that an organisation modelling approach would only apply in the hard-wired circulatory system case. However, as is shown in this paper, also in the free mobility living cell case, the organisation modelling approach can be used. Hence, not only a physically forced structure can be used and further analysed as an organisation structure, but also an organisational structure that emerges out of a number possibilities for interaction between processes can be successfully analysed.

Acknowledgements

The authors have learned a lot of this area from discussions and cooperation with Jaap Heringa, Jacky Snoep, Hans Westerhoff, Wouter Wijngaards.

References

Barringer, H., M. Fisher, D. Gabbay, R. Owens, and M. Reynolds (1996). The Imperative Future: Principles of Executable Temporal Logic, Research Studies Press Ltd. and John Wiley & Sons.

Bosse, T., Jonker, C.M., Treur, J. (2004). Organisation Modelling for the Dynamics of Complex Biological Processes. In: G. Lindemann, D. Moldt, M. Paolucci, B. Yu (eds.), *Proceedings of the International Workshop on Regulated Agent-Based Social Systems: Theories and Applications, RASTA'02*. Lecture Notes in AI, vol. 2934. Springer Verlag, 2004, pp. 92-112.

Ferber, J., (1999). *Multiagent Systems*. Addison Wesley.

Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organisations in multi-agent systems. In: *Proceedings of the Third International Conference on Multi-Agent Systems* (ICMAS'98), IEEE Computer Society Press, pp. 128-135.

Ferber, J., and Gutknecht, O. (2000). Operational Semantics of a role-based agent architecture. In: Jennings, N.R. & Lesperance, Y. (eds.) *Intelligent Agents VI*, Lecture Notes in AI, vol. 1757, Springer Verlag, 2000, pp. 205-217.

Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., (2001). Organization Models and Behavioural Requirements Specification for Multi-Agent Systems. In: Y. Demazeau, F. Garijo (eds.), *Multi-Agent System Organisations. Proceedings of the 10th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'01*, 2001. Lecture Notes in AI, Springer Verlag. To appear, 2002.

Jonker, C.M., Letia, I.A., and Treur, J., (2002). Diagnosis of the Dynamics within an Organisation by Trace Checking of Behavioural Requirements. In: Wooldridge, M., Weiss, G., and Ciancarini, P. (eds.), *Proc. of the 2nd International Workshop on Agent-Oriented Software Engineering, AOSE'01*. Lecture Notes in Computer Science, vol. 2222. Springer Verlag, 2002, pp. 17-32.

Jonker, C.M., Snoep, J.L., Treur, J., Westerhoff, H.V., and Wijngaards, W.C.A. (2002). The Living Cell as an Organisation: A Compositional Organisation Model of Intracellular Dynamics. Technical Report. Vrije Universiteit Amsterdam, Department of Artificial Intelligence, 2002. Also included as Ch. 7 in: Wijngaards, W.C.A., *Agent-Based Modelling of Dynamics: Biological and Organisational Applications.* Ph.D. Thesis. Vrije Universiteit Amsterdam, Department of Artificial Intelligence, 2002, pp. 189-254.

Jonker, C.M. and Treur, J. (1998). Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), Proceedings of the International Workshop on Compositionality, COMPOS'97. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380. Extended version in: *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

Jonker, C.M. and Treur, J. (2005). Agent-Oriented Modeling of the Dynamics of Complex Biological Processes I: Single Agent Models. *BioComplexity Journal*, vol. 1. In press, 2002

Jonker, C.M., Treur, J., and Wijngaards, W.C.A. (2002). Temporal Languages for Simulation and Analysis of the Dynamics Within an Organisation. In: B. Dunin-Keplicz and E. Nawarecki (eds.), *From Theory to Practice in Multi-Agent Systems, Proc. of the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems, CEEMAS'01*, 2001. Lecture Notes in AI, vol. 2296, Springer Verlag, 2002, pp. 151-160.

Kreitner, R., and Kunicki, A. (2001). Organisational Behavior, McGraw – Hill.

Lomi, A., and Larsen, E.R. (2001). Dynamics of Organizations: Computational Modeling and Organization Theories, AAAI Press, Menlo Park. Manna, Z., and Pnueli, A. (1995). *Temporal Verification of Reactive Systems: Safety*. Springer Verlag.

Mintzberg, H. (1979). The Structuring of Organisations, Prentice Hall, Englewood Cliffs, N.J.

Moss, S., Gaylard, H., Wallis, S., and Edmonds, B. (1998). SDML: A Multi-Agent Language for Organizational Modelling, *Computational and Mathematical Organization Theory 4*, (1), 43-70.

Neidhardt, F.C., Curtiss III, R., Ingraham, J.L., Lin, E.C.C., Brooks Low, K., Magasanik, B., Reznikoff, W.S., Riley, M., Schaechter, M., and Umbarger, H.E., eds. (1996). *Escherichia coli* and *Salmonella typhimurium*. ASM Press, Washington, D.C.

Prietula, M., Gasser, L., Carley, K. (1997). Simulating Organizations. MIT Press.

Takahashi, K. (2004). E-Cell: A Multi-Algorithm, Multi-Timescale Simulation Software Environment. URL: http://ecell.sourceforge.net/.

Weiss, G. (ed.) (1999). *Multiagent Systems*. MIT Press

PART VI REPRESENTATIONAL CONTENT

CHAPTER 16

Representational Content and the Reciprocal Interplay of Agent and Environment

This chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2005). Representational Content and the Reciprocal Interplay of Agent and Environment. In: Leite, J., Omincini, A., Torroni, P., and Yolum, P. (eds.), *Proceedings of the Second International Workshop on Declarative Agent Languages and Technologies, DALT'04*. Lecture Notes in Artificial Intelligence, vol. 3476, Springer Verlag, pp. 270-288.

An abstract of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2004). Representational Content and the Reciprocal Interplay of Agent and Environment (extended abstract). In: Jennings, N.R., Sierra, C., Sonenberg, L., and Tambe, M. (eds.), *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'04*. ACM Press, pp. 1408-1409.

A preliminary version of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2003). Representational Content and the Reciprocal Interplay of Agent and Environment. In: Sun, R. (ed.), *Proceedings of the IJCAI 2003 Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions*, pp. 19-28.

Representational Content and the Reciprocal Interplay of Agent and Environment

Tibor Bosse[1], Catholijn M. Jonker[1], and Jan Treur[1,2]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email : {tbosse, jonker, treur}@cs.vu.nl
URL : http://www.cs.vu.nl/~{tbosse, jonker, treur}
[2] Universiteit Utrecht, Department of Philosophy,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Abstract. Declarative modelling approaches in principle assume a notion of representation or representational content for the modelling concepts. The notion of representational content as discussed in literature in cognitive science and philosophy of mind shows complications as soon as agent and environment have an intense reciprocal interaction. In such cases an internal agent state is affected by the way in which internal and external aspects are interwoven during (ongoing) interaction. In this paper it is shown that the classical correlational approach to representational content is not applicable, but the temporal-interactivist approach is. As this approach involves more complex temporal relationships, formalisation was used to define specifications of the representational content more precisely. These specifications have been validated by automatically checking them on traces generated by a simulation model. Moreover, by mathematical proof it was shown how these specifications are entailed by the basic local properties.

1. Introduction

Declarative modelling approaches go hand in hand with some assumed notion of representation or representational content for the modelling concepts. Within cognitive and philosophical literature, classical approaches to representational content are based on correlations between an agent's internal state properties and external state properties. For example, the presence of a horse in the field is correlated to an internal state property that plays the role of a percept for this horse. One of the critical evaluations of this approach addresses the limitation that it is static: internal state properties are to be related to single external states, and cannot be related to processes involving multiple states or events over time. Especially in cases where the agent-environment interaction takes the form of an extensive reciprocal interplay in which both the agent and the environment contribute to the process in a mutual dependency, a classical approach to representational content is insufficient. Some authors even claim that it is a bad idea to aim for a notion of representation in such cases; e.g., [7], [12]. Therefore these cases can be considered a serious challenge to declarative methods.

As an alternative, within Philosophy of Mind, the *interactivist* approach [1] is put forward. In [5] it is shown how a temporal-interactivist approach to representational content of an internal state property can be formalised based on sets of agent-environment past and future interaction trajectories or traces.

In this paper it is analysed how some non-classical approaches may be used to define representational content in the case of an extensive agent-environment interplay. In particular, for a case study it will be discussed how the temporal-interactivist approach

and second-order approach to representational content can be used. These alternative notions involve more complex temporal relationships between internal and external states. Formalisation to define specifications of the representational content more precisely was used as a means to handle this complexity. This formalisation provided dynamic properties that can be (and actually have been) formally checked for given traces of the agent-environment interaction.

In Section 2 the modelling approach is briefly introduced. Section 3 introduces the case study and the language used to model this case study. In Section 4 a number of local dynamic properties describing basic mechanisms for the case study are presented; simulations on the basis of these local dynamic properties are discussed in Section 5. Section 6 presents global dynamic properties, describing the process as a whole and larger parts of the process. In Section 7 the interlevel relations between these nonlocal properties and the local properties are discussed. In Section 8 three different approaches to representational content are explored and formalised for the case study. In Section 9 it is shown how these formalisations can be validated against the simulation model, both by mathematical proof and by automated checks. Section 10 is a discussion.

2. Modelling Approach

To formally specify dynamic properties that express criteria for representational content from a temporal perspective an expressive language is needed. To this end the *Temporal Trace Language* is used as a tool; cf. [4]. In this paper for most of the occurring properties both informal or semi-formal and formal representations are given. The formal representations are based on the Temporal Trace Language (TTL), which is briefly defined as follows.

A *state ontology* is a specification (in order-sorted logic) of a vocabulary, i.e., a signature. A state for ontology Ont is an assignment of truth-values {true, false} to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP(Σ) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the input state of the organism (i.e., state of sensors for external world and body) at time point t is denoted by state(γ, t, input); analogously, state(γ, t, output), state (γ, t, internal) and state (γ, t, EW) denote the output state, internal state and external state (of the world, including the physical body) for the organism.

These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus (see [11] for an introduction, and [10] for an example application): state(γ, t, output) |= p denotes that state property p holds in trace γ at time t in the output state of the organism. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts T for time points, Traces for traces and F for state

formulae, using quantifiers over time and the usual first-order logical connectives such as $\neg, \wedge, \vee, \Rightarrow, \forall, \exists$.

To model direct temporal dependencies between two state properties, the simpler *leads to* format is used. This is an executable format defined as follows. Let α and β be state properties of the form "conjunction of literals" (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \rightarrow\!\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h .

For a precise definition of the *leads to* format in terms of the language TTL, see [6]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically. The *leads to* format has shown its value especially when temporal or causal relations in the (continuous) physical world are modelled and simulated in an abstract, non-discrete manner; for example, the intracellular chemistry of *E. coli* [3].

3. The Case Study

In this Section the case study will be introduced and the internal state properties and their dynamics to model this example are presented.

3.1. Introduction of the Case Study

The case study addressed involves the processes to unlock a front door that sticks. Between the moment that the door is reached and the moment that the door unlocks the following reciprocal interaction takes place:

- the agent puts rotating pressure on the key,
- the door lock generates resistance in the interplay,
- the agent notices the resistance and increases the rotating pressure,
- the door increases the resistance,
- and so on, without any result.
- finally, after noticing the impasse the agent changes the strategy by at the same time pulling the door and turning the key, which unlocks the door.

This example shows different elements. The first part of the process is described in terms of Sun's sub-conceptual level, whereas the last part of the process is viewed in terms of the conceptual level [12], [13]. For both parts of the process the notion of representational content will be discussed and formalised.

3.2. State Properties

To model the example the following internal state properties are used:

s1	sensory representation for being at the door
s2(r)	sensory representation for resistance r of the lock
p1(p)	preparation for the action to turn the key with rotating pressure p (without pulling the door)
p2	preparation for combined pulling the door and turning the key
c	state for having learnt that turning the key should be combined with pulling the door

The interactions between agent and environment are defined by the following sensor and effector states:

o1	observing being at the door
o2(r)	observing resistance r
a1(p)	action turn the key with rotating pressure p (without pulling the door)
a2	action turn the key while pulling the door

In addition, the following state properties of the world are used:

arriving_at_door	the agent arrives at the door
lock_reaction(r)	the lock reacts with resistance r
door_unlocked	the door is unlocked
d(mr)	resistance threshold mr of the door (indicating that the door will continue to resist until pressure mr or more is used)
max_p(mp)	maximal force on the key that can be exercised by the agent.

4. Local Dynamic Properties

To model the dynamics of the example, the following local properties (in *leads to* format) are considered. They describe the basic parts of the process.

LP1 (observation of door)
The first local property LP1 expresses that the world state property arriving_at_door leads to an observation of being at the door. Formalisation:
arriving_at_door $\twoheadrightarrow$ o1

LP2 (observation of resistance)
Local property LP2 expresses that the world state property lock_reaction with resistance r leads to an observation of this resistance r.
lock_reaction(r) $\twoheadrightarrow$ o2(r)

Note that r is a variable here; the specification should be read as a schema for the set of all instances for r.

LP3 (sensory representation of door)
Local property LP3 expresses that the observation of being at the door leads to a sensory representation for being at the door.
o1 $\twoheadrightarrow$ s1

LP4 (sensory representation of resistance)
LP4 expresses that the observation of resistance r of the lock leads to a sensory representation for this resistance.
o2(r) $\twoheadrightarrow$ s2(r)

LP5 (action preparation initiation)
LP5 expresses that a sensory representation for being at the door leads to a preparation for the action to turn the key with pressure 1.
s1 $\twoheadrightarrow$ p1(1)

LP6 (pressure adaptation)
LP6 expresses the following: if turning the key with a certain pressure p did not succeed (since the agent received a resistance that equals p), and the agent has not reached its maximal force (p<mp), and the agent has not learnt anything yet (not c), then it will increase its pressure.
p1(p) and s2(r) and p=r and p<mp and not c $\twoheadrightarrow$ p1(p+1)

LP7 (birth of learning state)
LP7 expresses that, if turning the key with a certain pressure p did not succeed (since the agent received a resistance that equals p), and the agent has reached the limit of its force (p≥mp), then it will learn that should perform a different action.
p1(p) and s2(r) and p=r and p≥mp $\twoheadrightarrow$ c

LP8 (learning state persistency)
LP8 expresses that the learning state property c persists forever.
c $\twoheadrightarrow$ c

LP9 (alternative action preparation)
LP9 expresses that a sensory representation for resistance r of the lock together with the learning state property lead to a preparation for combined pulling of the door and turning the key.
c and s2(r) $\twoheadrightarrow$ p2

LP10 (action performance)
LP10 expresses that a preparation for the action to turn the key with pressure p (without pulling the door) leads to the actual performance of this action.
p1(p) $\twoheadrightarrow$ a1(p)

LP11 (alternative action performance)
LP11 expresses that a preparation for combined pulling of the door and turning the key leads to the actual performance of this action.
p2 $\twoheadrightarrow$ a2

LP12 (negative effect of action)
LP12 expresses the following property of the world: if the key is turned with a certain pressure p that is smaller than the maximal resistance of the door (p<mr), and the agent is not pulling the door simultaneously, then the lock will react with resistance p.
a1(p) and not a2 and d(mr) and p<mr $\twoheadrightarrow$ lock_reaction(p)

LP13 (positive effect of action)
LP13 expresses the following property of the world: if the key is turned with a certain pressure p that at least equals the maximal resistance of the door (p≥mr), then the door will unlock.
a1(p) and d(mr) and p≥mr $\twoheadrightarrow$ door_unlocked

LP14 (positive effect of alternative action)
LP14 expresses the following property of the world: if the agent turns the key, and simultaneously pulls the door, then the door will unlock.
a2 $\twoheadrightarrow$ door_unlocked

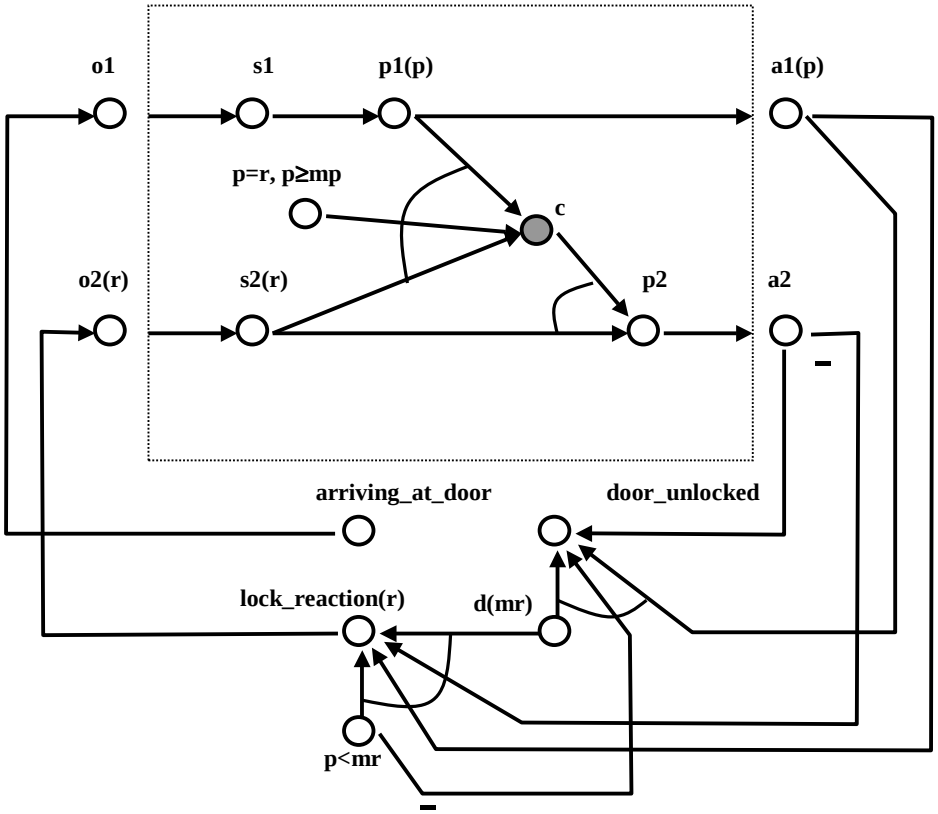


Figure 1. Overview of the simulation model

In Figure 1 an overview of these properties is given in a graphical form. To limit complexity, local property LP6 is not depicted.

5. Simulation

A special software environment has been created to enable the simulation of executable models. Based on an input consisting of dynamic properties in *leads to* format, the software environment generates simulation traces. An example of such a trace can be seen in Figure 2. Time is on the horizontal axis, the state properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false. This trace is based on all local properties identified above. In property LP6, the values (0,0,1,5) have been chosen for the timing parameters e, f, g, and h. In all other properties, the values (0,0,1,1) have been chosen.

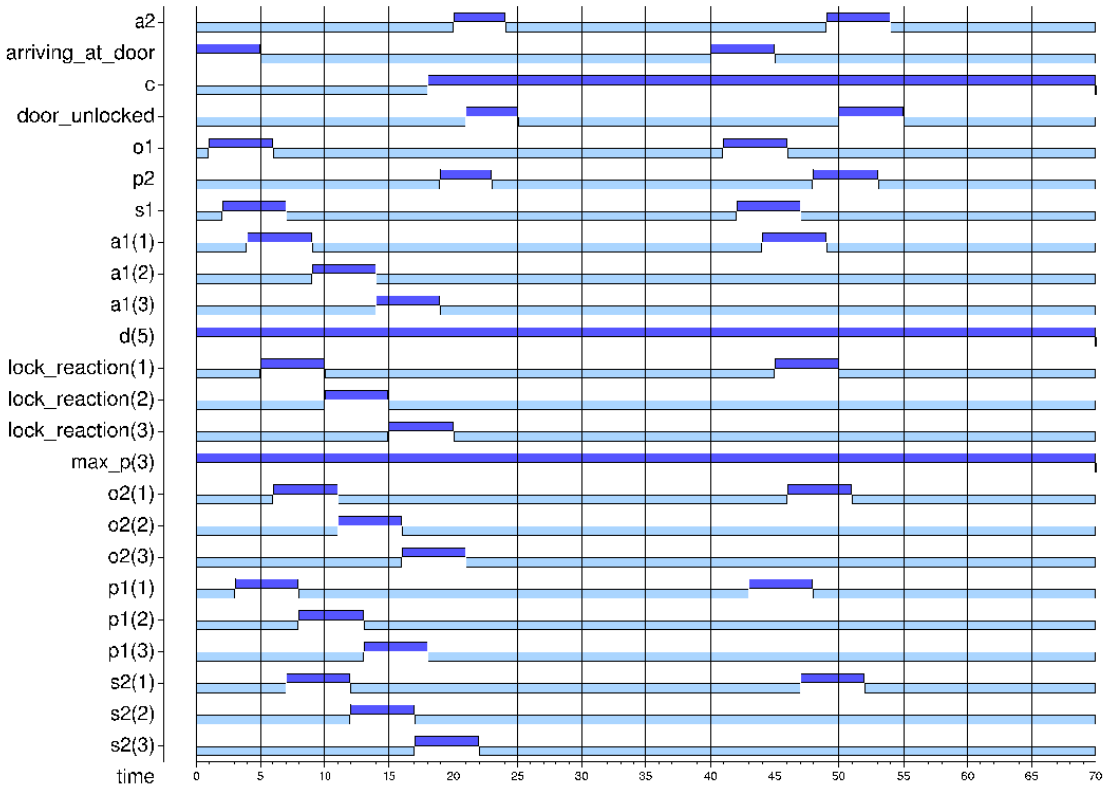


Figure 2. Example simulation trace

As can be seen in Figure 2, the presence of the agent at the door leads to a corresponding observation result (o1), followed by a sensory representation for being at the door. Next, the agent prepares for turning the key (initially with pressure 1), and subsequently performs this action. Since this pressure is insufficient to unlock the door (within this example, the resistant threshold of the door is 5), the door does not open, but a lock reaction (with resistance 1) occurs instead. As a consequence, the agent observes this resistance, and creates a sensory representation of it. At this point, the agent prepares to increase the pressure (see local property LP6), resulting in the action of turning the key with pressure 2. This loop is being activated once more: the agent even tries to turn the key with pressure 3, but then reaches the limit of its force (3 in this example, see LP7) and learns that it should perform a different action. In other words, internal state property c

becomes true. Subsequently, the combination of this state property c and state property s2(3) leads to the preparation for an alternative action: combined pulling of the door and turning the key. As a result of this preparation, the action is actually performed and the door is unlocked. After that, to show that the agent has indeed learned something, the trace continues for a while. At time point 40, the agent again finds itself confronted with a locked door. Again, it starts by trying to turn the key with pressure 1. However, when this approach turns out not to work, this time the agent shows adapted behaviour. It does not try to increase the pressure, but immediately switches to the alternative action instead.

6. Non-local Dynamic Properties

This section presents dynamic properties for larger parts of the process, i.e., at a nonlocal level. Within these properties, γ is a variable that stands for an arbitrary trace.

GP1 (door eventually unlocked)
Global property GP1 expresses that eventually the door will be unlocked.

$\forall t$: state(γ, t, EW) $\models$ arriving_at_door $\Rightarrow$
  $\exists t' \geq t$: state(γ, t', EW) $\models$ door_unlocked

GP2 (learning occurs)
Global property GP2 expresses that if the maximal resistance of the door is bigger than the maximal rotation force that the agent can exert, then at some point in time learning will occur.

$\forall t$: state(γ, t, EW) $\models$ d(mr) $\wedge$
  $\forall t$: state(γ, t, internal) $\models$ max_p(mp) $\wedge$ mr > mp $\Rightarrow$
    $\exists t'$ state(γ, t', internal) $\models$ c

GP3 (mr > mp $\Rightarrow$ door eventually unlocked)
Global property GP3 expresses that if the maximal resistance of the door is bigger than the maximal rotation force that the agent can exert, then at some point in time the door will be unlocked.

$\forall t$: state(γ, t, EW) $\models$ d(mr) $\wedge$
  $\forall t$: state(γ, t, internal) $\models$ max_p(mp) $\wedge$ mr > mp $\Rightarrow$
    $\exists t'$ state(γ, t', EW) $\models$ door_unlocked

GP4 (mr $\leq$ mp $\Rightarrow$ door eventually unlocked)
Global property GP4 expresses that if the maximal resistance of the door is less than or equal to the maximal rotation force that the agent can exert, then at some point in time the door will be unlocked.

$\forall t$: state(γ, t, EW) $\models$ d(mr) $\wedge$
  $\forall t$: state(γ, t, internal) $\models$ max_p(mp) $\wedge$ mr $\leq$ mp $\Rightarrow$
    $\exists t'$ state(γ, t', EW) $\models$ door_unlocked

GP3 and GP4 are formulated separately because their proofs differ. Next a number of intermediate properties are formulated that form a kind of milestones in the process of opening a door and learning.

M1 (at door $\Rightarrow$ preparation to turn key)
Intermediate property M1 expresses that after the agent stands at the door the agent will prepare for turning the key.

$\forall t$: state(γ, t, EW) $\models$ arriving_at_door $\Rightarrow$
  $\exists t' > t$: state(γ, t', internal) $\models$ p1(1)

M2 (lock reaction represented)
Intermediate property M2 expresses that a lock reaction will be represented internally.

$\forall t$: state(γ, t, EW) $\models$ lock_reaction(r) $\Rightarrow$
  $\exists t' > t$: state(γ, t', internal) $\models$ s2(r)

M3 (alternative action)
M3 expresses that if lock resistance is internally represented and the agent has learned, then at some later point in time the agent will perform the action a2.
$\forall t$: state(γ, t, internal) $\models$ c $\wedge$ state(γ, t, internal) $\models$ s2(r) $\Rightarrow$
$\exists t' > t$: state(γ, t, ouput) $\models$ a2

M4 (increasing rotation pressure)
M4 expresses that under the condition that agent has not learned c yet, the rotation pressure that the agent exerts on the key will always reach the minimum of the maximal resistance of the door and the maximal force that the agent can exert.
$\forall t, \forall mp, \forall mr, \forall sl$
not state(γ, t, internal) $\models$ c $\wedge$ state(γ, t, EW) $\models$ d(mr) $\wedge$
state(γ, t, internal) $\models$ max_p(mp) $\wedge$ sl = minimum(mr, mp) $\wedge$
state(γ, t, EW) $\models$ arriving_at_door $\Rightarrow$
$\exists t' > t$: state(γ, t', internal) $\models$ p1(sl) $\wedge$ $\exists t'' > t'$: state(γ, t'', output) $\models$ a1(sl)

Finally, a number of additional properties are needed in order to prove the relations between the properties.

A1 (maximal force)
Additional property A1 expresses that the maximal rotation force that the agent can exert on the key is constant.
$\exists mp \forall t$: state(γ, t, internal) $\models$ max_p(mp)

A2 (maximal resistance)
Additional property A2 expresses that the maximal resistance that the door can offer is constant.
$\exists mr \forall t$: state(γ, t, EW) $\models$ d(mr)

A3 (Closed World Assumption)
The second order property that is commonly known as the Closed World Assumption expresses that at any point in time a state property that is not implied by a specification to be true is false. Let Th be the set of all local properties LP1-LP14.
$\forall P \in$ At(ONT) $\forall t$: not Th |-- state(γ, t) $\models$ P $\Rightarrow$ state(γ, t) $\models$ not P

7. Interlevel Relations

This section outlines the interlevel connections between dynamic properties at different levels, varying from dynamic properties at the local level of basic parts of the process to dynamic properties at the global level of the overall process. The following interlevel relations between local dynamic properties and non-local dynamic properties can be identified.

GP3 & GP4	$\Rightarrow$ GP1
M2 & M4 & LP7 & LP12	$\Rightarrow$ GP2
M2 & M3 & M4 & LP7 & LP14	$\Rightarrow$ GP3
M4 & LP13	$\Rightarrow$ GP4
LP1 & LP3 & LP5	$\Rightarrow$ M1
LP2 & LP4	$\Rightarrow$ M2
LP8 & LP9 & LP11	$\Rightarrow$ M3
M1 & M2 & LP6 & LP10 & LP12 & A1 & A2 & A3	$\Rightarrow$ M4

The proofs of M1, M2, M3, and GP1 are rather straightforward and left out. A proof sketch of the other properties is provided.

Property M4 can be proved by induction. The induction step is

$\forall t$: state(γ, t, output) $\models$ a1(p) $\wedge$ p < sl $\Rightarrow$
$\exists t1 > t, \exists t2 > t1$:
state(γ, t1, internal) $\models$ p1(p+1) $\wedge$ state(γ, t2, output) $\models$ a1(p+1)

The induction base is given by properties M1 and LP10, providing p1(1), and a1(1). The induction step is proved along the following lines.

- "not a2" holds at all times during which "not c" holds.

This is proved on the basis of "not c" and A3. A3 states that if a2 cannot be derived from the specification at a certain point in time, then "not a2" holds at that time. So pick any point in time at which "not c" holds and try to prove a2 from all local properties and the additional assumptions A1, A2, and A3. If a2 can be proven, it is due to LP11. The condition of LP11 is p2. The only way to prove p2 is through LP9. The conditions of LP9 are c and s2(r). The condition c is in direct contradiction with "not c". In the above the temporal elements of the proof were not mentioned. To complete this proof these elements do play a role, for example, c cannot change its truth more than once. It starts out false and remains false until (by application of LP7) it becomes true. Once c is true, it remains true by application of LP8. Therefore, as long as "not c" holds, "not a2" also holds (and even a bit longer).

- a1(p) holds

In proving the induction step, the condition is assumed. Thus a1(p) holds.

- d(mr) holds

Direct from A2.

- p < mr

This is true, since p < sl and sl is the minimum of mp and mr.

- lock_reaction(p) holds.

All conditions of LP12 hold (i.e., a1(p), not a2, d(mr), p < mr), thus LP12 can be applied, which makes sure that lock_reaction(p) holds some time later.

- s2(p) holds

Based on lock_reaction(p), M2 can be applied, thus some time later s2(p) will hold.

- p1(p+1) holds at some time point t1 later than the chosen time t.

By application of LP6 some time later (call this time point t1) p1(p+1) will hold . Note that the conditions of LP6 are met: p < mp holds, since p < sl, and sl the minimum of mp and mr.

- a1(p+1) holds at some time t2 later than t1.

This is proved by applying LP10 with p+1. This proves that the induction step holds.

Now assuming that the antecedent of M4 holds, implies that subsequently (over time) LP1, LP3, LP5 and LP10 can be applied. In that manner, from arriving at the door, an observation of that fact is derived, leading to an internal representation thereof (s1), leading to an internal state in which p1(1) holds, leading to an output state in which a1(1) holds. Therefore, all circumstances hold for the induction step to be applicable. Application of the induction step leads to the conclusion that at some point in time p1(sl) holds in the internal state and some time later again a1(sl) holds in the output state. Thus proving the conclusion of M4 under the assumption that the antecedent of M4 holds. This concludes the proof by induction of M4.

Property GP2 can be proved as follows. Since mr > mp, sl = mp. Applying M4 gives us $\exists t'$: state(γ, t', output) $\models$ a1(mp). By application of LP12, we get some time later lock_reaction(mp), application of M2 gives us, some time later again, s2(mp). Finally, application of LP7 provides us with the learned c.

The proof of Property GP3 follows the following subsequent time points of interest: application of M4 gives a time point t1 such that p1(mp) holds, application of M2 gives a time t2 such that s2(mp) holds, application of LP7 gives a time t3 such that c holds, application of M3 gives a time t4 such that a2 holds, application of LP14 gives a time t5 such that door_unlocked holds.

The proof of property GP4 is rather short, by application of M4 at a certain time t1 a1(mr) will hold, by application of LP13 a later time t2 exist at which door_unlocked holds. All proofs can be worked out in more details by using the timing parameters of the local properties involved.

8. Representational Content

In the literature on Philosophy of Mind different types of approaches to representational content of an internal state property have been put forward, for example the correlational, interactivist, relational specification and second-order representation approach; cf. [8], pp. 191-193, 200-202, [1]. These approaches have in common that the occurrence of the internal state property at a specific point in time is related to the occurrence of other state properties, at the same or at different time points. The "other state properties" can be of three types:

A) external world state properties, independent of the agent

B) the agent's sensor state and effector state properties, i.e., the agent's interaction state properties (interactivist approach)

C) internal state properties of the agent (higher-order representation)

Furthermore, the type of relationships can be (1) purely functional *one-to-one correspondences*, (e.g., the correlational approach), or (2) they can involve more *complex relationships* with a number of states at different points in time in the past or future, (e.g., the interactivist approach). So, six types of approaches to representational content are distinguished, that can be indicated by codings such as A1, A2, and so on. Below, examples are given.

8.1. Correlational Approach

According to the Correlational approach, the representational content of a certain internal state is given by a one-to-one correlation to another (in principle external) state property: type A1. Such an external state property may exist backward as well as forward in time. Hence, for the current example, the representational content for internal state property s1 can be defined as world state property arriving_at_door, by looking backward in time. Intuitively, this is a correct definition, since for all possible situations where the agent has s1, it was indeed physically present at the door, and conversely. Likewise, the representational content for internal state property p2 can be defined as action property a2, by looking forward in time, or, rather, as world state property door_unlocked. However, for many other internal state properties the representational content cannot be defined adequately according to the correlational approach. In these cases, reference should not be made to one single state in the past or in the future, but to a temporal sequence of inputs or output state properties, which is not considered to adequately fit in the correlational approach. An overview for the content of all internal state properties according to the correlational approach (if any), is given in Table 1. These relationships can easily be specified in the language TTL.

Internal state property	Content (backward)	Content (forward)
s1	arriving_at_door	lock_reaction(1)
s2(r)	lock_reaction(r)	*impossible*
p1(1)	arriving_at_door	lock_reaction(1)
p1(2)	*impossible*	lock_reaction(2)
p2	*impossible*	door_unlocked
c	*impossible*	*impossible*

Table 1. Correlational approach

8.2. Temporal-Interactivist Approach

The temporal-interactivist approach [1], [5] relates the occurrence of internal state properties to sets of past and future interaction traces: type B. This can be done in the form of functional one-to-one correspondences (type B1), or by involving more complex relationships over time (type B2). In this paper the focus is on the more advanced case, i.e., the B2 type. As an example, consider the internal state property c. The representational content of c is defined in a semantic manner by the pair of sets of past interaction traces and future interaction traces (here InteractionOnt denotes the input and output state ontology and IntOnt the internal state ontology; $\gamma_{\leq t}^{InteractionOnt}$ denotes the trace γ up to t, with states restricted to the interaction states):

$$PITRACES(c) = \{ \gamma_{\leq t}^{InteractionOnt} \mid t \in T, state(\gamma, t, IntOnt) \models c\}$$
$$FITRACES(c) = \{ \gamma_{\geq t}^{InteractionOnt} \mid t \in T, state(\gamma, t, IntOnt) \models c \}$$

Here the first set, PITRACES(c), contains all past interaction traces for which sequence of time points exists such that at these time points first o1 occurs, next a1(1), next o2(1), next a1(2), next o2(2), next a1(3), and next o2(3). For this example, a learning phase of 3 trials has been chosen. The second set, FITRACES(c), contains all future interaction traces for which no o2(r) occurs, or o2(r) occurs and after this a2 occurs.

An overview for the representational content of all internal state properties according to the temporal-interactivist approach is given, in an informal notation, in Table 2.

I.s.p.	Content (backward)	Content (forward)
s1	o1	a1(1)
s2(r)	o2(r)	if c (defined by o1, …, o2(3)), then a2
p1(1)	o1	a1(1)
p1(2)	o1, a1(1), o2(1)	a1(2)
p1(3)	o1, a1(1), o2(1), a1(2), o2(2)	a1(3)
p2	o1, a1(1), o2(1), a1(2), o2(2), a1(3), o2(3)	a2
c	o1, a1(1), o2(1), a1(2), o2(2), a1(3), o2(3)	if o2(r), then a2

Table 2. Temporal-interactivist approach (semantic description)

Note that these relationships are defined at a semantic level, and are thus of type B2a. Different interaction state properties, separated by commas, should be read as the temporal sequence of these states. Again, a learning phase of 3 trials has been chosen. In order to obtain a description at a syntactic level, the relationships given in Table 2 are characterised by formulae in a specific language, TTL in our case. Thus, the representational content of a certain internal state is then defined by specifying a formal temporal relation of the internal state property to sensor and action states in the past and future. A number of such formal temporal relations are given in Table 3. Because of space limitations, only the backward content is shown.

I.s.p.	Content (backward)
s1	is_followed_by(γ, o1, input, s1, internal) & is_preceded_by(γ, s1, internal, o1, input)
s2(r)	is_followed_by(γ, o2(r), input, s2(r), internal) & is_preceded by(γ, s2(r), internal, o2(r), input)
p1(1)	is_followed_by(γ, o1, input, p1(1), internal) & is_preceded by(γ, p1(1), internal, o1, input)
p1(2)	∀t1,t2,t3 [t1≤t2≤t3 & state(γ, t1, input) ⊨ o1 & interplay_up_to(γ, t2, t3,1) & not [∃t11,t12,t17 [t11≤t12≤t17≤t3 & state(γ, t11, input) ⊨ o1 & interplay_up_to(γ, t12, t17,3)]] ⇒ ∃t4 ≥ t3 state(γ, t4, internal) ⊨ p1(2)] & ∀t4 [state(γ, t4, internal) ⊨ p1(2) ⇒ ∃t1,t2,t3 t1≤t2≤t3≤t4 & state(γ, t1, input) ⊨ o1 & interplay_up_to(γ, t2, t3,1)]
p1(3)	∀t1,t2,t5 [t1≤t2≤t5 & state(γ, t1, input) ⊨ o1 & interplay_up_to(γ, t2, t5, 2) ⇒ ∃t6 ≥ t5 state(γ, t6, internal) ⊨ p1(3)] & ∀t6 [state(γ, t6, internal) ⊨ p1(3) ⇒ ∃t1,t2,t5 t1≤t2≤t5≤t6 & state(γ, t1, input) ⊨ o1 & interplay_up_to(γ, t2, t5,2)]
p2	∀t1,t2,t7 [t1≤t2≤t7 & state(γ, t1, input) ⊨ o1 & interplay_up_to(γ, t2, t7,3) ⇒ ∃t8 ≥ t7 state(γ, t8, internal) ⊨ p2] & ∀t8 [state(γ, t8, internal) |== p2 ⇒ ∃t1,t2,t7 t1≤t2≤t7≤t8 & state(γ, t1, input) ⊨ o1 & interplay_up_to(γ, t2, t7,3)]
c	∀t1,t2,t7 [t1≤t2≤t7 & state(γ, t1, input) ⊨ o1 & interplay_up_to(γ, t2, t7,3) ⇒ ∃t8 ≥ t7 state(γ, t8, internal) ⊨ c] & ∀t8 [state(γ, t8, internal) ⊨ c ⇒ ∃t1,t2,t7 t1≤t2≤t7≤t8 & state(γ, t1, input) ⊨ o1 & interplay_up_to(γ, t2, t7,3)]

Table 3. Temporal-interactivist approach (syntactic description, backward)

Within Table 3, the following abstractions are used:

is_followed_by(γ, X, I1, Y, I2) ≡
  ∀t1: state(γ, t1, I1) ⊨ X ⇒ ∃t2 ≥ t1: state(γ, t2, I2) ⊨ Y

This expresses that X is always followed by Y.

is_preceded by(γ, Y, I1, X, I2) ≡
  ∀t1: state(γ, t2, I1) ⊨ Y ⇒ ∃t1 ≤ t2: state(γ, t1, I2) ⊨ X

This expresses that Y is always preceded by X. These abstractions can be used like is_preceded_by(γ, s1, internal, o1, input), is_followed_by(γ, o2(1), input, s2(1), internal), et cetera. The next abstraction describes that the interplay between agent and environment in which the agent increases pressure and the environment increases resistance is performed up to a certain level of pressure.

interplay_up_to(γ, t1, t2, 1) ≡ t1≤ t2 &
  state(γ, t1, output) ⊨ a1(1) & state(γ, t2, input) ⊨ o2(1)

interplay_up_to(γ, t1, t4, 2) ≡ ∃t2, t3 [t1 ≤ t2 ≤ t3 ≤ t4]
  interplay_up_to(γ, t1, t2, 1) &
  state(γ, t3, output) ⊨ a1(2) & state(γ, t4, input) ⊨ o2(2)

interplay_up_to(γ, t1, t6, 3) ≡ ∃t4, t5 [t1 ≤ t4 ≤ t5 ≤ t6]
  interplay_up_to(γ, t1, t4, 2) &
  state(γ, t5, output) ⊨ a1(3) & state(γ, t6, input) ⊨ o2(3)

8.3. Second-Order Representation

In approaches to representational content of type C, internal state properties are related to other internal state properties. For example, in Sun's dual approach to cognition [12], [13], conceptual level state properties are related to subconceptual level state properties:

> *On this view, high-level conceptual, symbolic representation is rooted, or grounded, in low-level behavior (comportment) from which it obtains its meanings and for which it provides support and explanations. The rootedness/groundedness is guaranteed by the way high-level representation is produced: It is, in the main, extracted out of low-level behavioral structures. (Sun, 2000).*

Two possibilities arise: either the other internal state properties are not considered to be representational (this seems to be Sun's position), or they are themselves considered representations of something else. In the latter case, which is explored here, the conceptual level state properties become second-order representations: representations of representations. In the main example of this paper, the internal state property c can be considered to be at the conceptual level, whereas the other, s and p properties are considered subconceptual. Then, in the spirit of [12], the representational content of c can be defined in terms of the other internal state properties as shown below. However, keep in mind that this approach only makes sense if the low-level internal state properties are considered to be representational already.

Backward: c will occur if in the past once s1 occurred, then p1(1), then s2(1), then p1(2), then s2(2), then p1(3), then s2(3), and conversely. Formally:

$\forall t1,t2,t3,t4,t5,t6,t7$ [$t1 \leq t2 \leq t3 \leq t4 \leq t5 \leq t6 \leq t7$
& state(γ, t1, internal) $\models$ s1
& state(γ, t2, internal) $\models$ p1(1) & state(γ, t3, internal) $\models$ s2(1)
& state(γ, t4, internal) $\models$ p1(2) & state(γ, t5, internal) $\models$ s2(2)
& state(γ, t6, internal) $\models$ p1(3) & state(γ, t7, internal) $\models$ s2(3)
$\Rightarrow \exists t8 \geq t7$ state(γ, t8, internal) $\models$ c] &
$\forall t8$ [state(γ, t8, internal) $\models$ c $\Rightarrow$
$\exists t1,t2,t3,t4,t5,t6,t7$ $t1 \leq t2 \leq t3 \leq t4 \leq t5 \leq t6 \leq t7 \leq t8$
& state(γ, t1, internal) $\models$ s1
& state(γ, t2, internal) $\models$ p1(1) & state(γ, t3, internal) $\models$ s2(1)
& state(γ, t4, internal) $\models$ p1(2) & state(γ, t5, internal) $\models$ s2(2)
& state(γ, t6, internal) $\models$ p1(3) & state(γ, t7, internal) $\models$ s2(3)]

Forward: if c occurs, then in the future, if s2(r) occurs, then p2 will occur. Formally:

$\forall t1$ [state(γ, t1, internal) $\models$ c $\Rightarrow$
$\forall t2 \geq t1$ [state(γ, t2, internal) $\models$ s2(r) $\Rightarrow$
$\exists t3 \geq t2$ state(γ, t3, internal) $\models$ p2]]

9. Validation

A large variety of techniques exist for (automated) verification of relevant properties of complex systems, for examples see [9], [14], [16] and the references in these papers. In the current research, the specifications of representational content have been validated in two ways: (1) by relating them to the local dynamic properties by mathematical proof, and (2) by automatically checking them for the simulated traces.

An example of the former is as follows. Consider the formula that presents the backward representational content for internal state property c in Table 3. Consider first the direction from observations to c. Given o1, o2(1), o2(2), and o2(3) at the different subsequent time points the proof obligation is c. Given o1, by applying (in this order) LP3, LP5 we obtain p1(1) which we need to derive from the given o2(1) using LP4, s2(1) and by application of LP6 on p1(1) and s2(1) we obtain p1(2). Given o2(2), by application

of LP4 we obtain s2(2) and on the basis of p1(2) LP6 is again applicable resolving into p1(3). Given o2(3), apply LP4 to obtain s2(3), and using p1(3) LP7 is applicable and c is obtained. These dependencies are graphically represented in Figure 3. The reverse direction again depends on property A3 and all local properties.

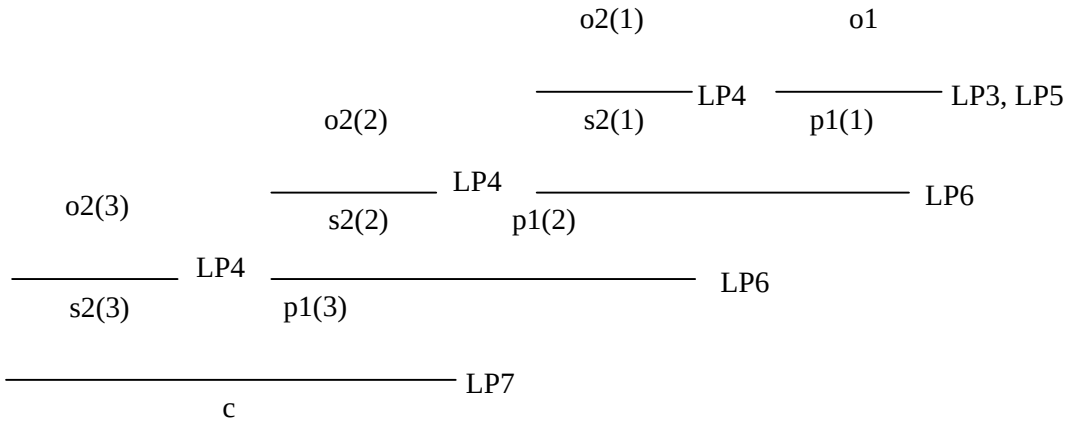


Figure 3. Proof Tree

In addition to the software described in Section 5, other software has been developed that takes traces and formally specified properties as input and checks whether a property holds for a trace. Using automatic checks of this kind, many of the properties presented in this paper have been checked against a number of generated traces as depicted in Figure 2. In particular, the global properties GP1, GP2, GP3, and GP4, and the intermediate properties M1, M2, M3, and M4 have been checked, and all turned out to hold for the given traces. Furthermore, all properties for representational content denoted in Table 3 have been checked. The duration of these checks varied from one second to a couple of minutes, depending on the complexity of the formula (in particular, the amount of time points). Success of these checks would validate our choice for the representational content (according to the temporal-interactivist approach) of the internal state properties s1, s2(r), p1(1), p1(2), p1(3), p2, and c. However, note that these checks are only an empirical validation, they are no exhaustive proof as, e.g., model checking is. Currently, the possibilities are explored to combine TTL with existing model checking techniques.

Although they are not exhaustive, even the empirical checks mentioned above have already proved their value. Initially, one of these checks did not succeed. It turned out that the backward representational content defined for p1(2) was not correctly chosen. At that time, it was defined as follows:

$$\forall t1,t2,t3\ [\ t1 \le t2 \le t3\ \&\ state(\gamma, t1, input) \models o1\ \&$$
$$interplay\_up\_to(\gamma, t2, t3, 1)$$
$$\Rightarrow \exists t4 \ge t3\ state(\gamma, t4, internal) \models p1(2)\ ]$$
$$\&\ \forall t4\ [\ state(\gamma, t4, internal) \models p1(2) \Rightarrow \exists t1,t2,t3\ \ t1 \le t2 \le t3 \le t4\ \&$$
$$state(\gamma, t1, input) \models o1\ \&\ interplay\_up\_to(\gamma, t2, t3, 1)\ ]$$

According to the above notation, the sequential occurrence of the state properties o1, a1(1), and o2(1) always implies that state property p1(2) will occur. However, a close examination of Figure 2 reveals that this is not always the case. Whenever the agent has learned, it will not increase its pressure on the key anymore. As a result, the extra

condition not c had to be added to the representational content. All the other checks concerning the properties of Table 3 did succeed immediately.

10. Discussion

The classical correlational approach to representational content requires a one-to-one correspondence between an internal state property of an agent and one external world state property. For embodied agents that have an extensive reciprocal interaction with their environment, this classical correlational approach does not suffice. In particular, an internal state in such an agent does not depend on just one state property of the external world, but is affected both by external aspects of the world and by internal aspects of the agent itself and the way in which these aspects are interwoven during the (ongoing) interaction process.

Given this problem, it is under debate among several authors whether adequate alternative notions of representational content exist for such an embodied agent's internal states. Some authors claim that for at least part of the internal states it makes no sense to consider them as conceptual or as having representational content; e.g., [2], [7], [11]. Other authors claim that some notions of representational content can be defined, but these strongly deviate from the classical correlational approach; e.g., [1], [5], [8].

Given the above considerations, the case of an intensive agent-environment interaction is a challenge for declarative approaches in the sense that internal states depending on such an interaction have no simple-to-define representational content. The formally defined and validated representation relations presented in this paper show how it is still possible to obtain a declarative perspective also for such a case. It is shown how formal methods allow to address the temporal structure entailed by suitable representation relations in these cases in a manageable declarative form.

More specifically, in this paper, for some notions of representational content it was explored in a case study how they work out, and, especially, how the temporal structure can be handled by formalisation. The processes of the case study have been formalised by identifying executable local dynamic properties for the basic dynamics. On the basis of these local properties a simulation model has been made. The formalised specifications of the representational content of the internal state properties have been validated by automatically checking them on the traces generated by the simulation model. Moreover, by mathematical proof it was shown how these specifications are entailed by the basic local properties. This shows that the internal state properties indeed fulfil the representational content specification.

The use of the temporal trace language TTL has a number of practical advantages. In the first place, it offers a welldefined language to formulate relevant dynamic relations in practical domains, with first order logic expressivity and semantics. Furthermore, it has the possibility of explicit reference to *time points* and *time durations* that enables modelling of the dynamics of continuous real-time phenomena, such as sensory and neural activity patterns in relation to mental properties. These features go beyond the expressive power available in standard linear or branching time temporal logics, such as LTL and CTL.

Moreover, the possibility to quantify over traces allows for specification of *more complex adaptive behaviours*. As within most temporal logics, reactiveness and pro-activeness properties are specified. In addition, in TTL also properties expressing different types of adaptive behaviour can be expressed. For example a property such as "exercise improves skill", which is a relative property in the sense that it involves the comparison of two alternatives for the history. Another property of this type is trust

monotony: “the better the experiences with something or someone, the higher the trust”. This type of relative property can be expressed in our language, whereas in standard forms of temporal logic different alternative histories cannot be compared. For an excellent review of standard temporal logics, see [15].

Note that, in addition to simulated traces, the TTL checking software is also able to take other (e.g., empirical) traces as input, enabling the validation of the representational content of internal states in real-world situations.

References

[1] Bickhard, M.H., Representational Content in Humans and Machines. *Journal of Experimental and Theoretical Artificial Intelligence*, 5, 1993, pp. 285-333.

[2] Clark, A., Being There: Putting Brain, Body and World Together Again. MIT Press, 1997.

[3] Jonker, C.M., Snoep, J.L., Treur, J., Westerhoff, H.V., and Wijngaards, W.C.A., BDI-Modelling of Intracellular Dynamics. In: A.B. Williams and K. Decker (eds.), *Proc. of the First International Workshop on Bioinformatics and Multi-Agent Systems, BIXMAS'02*, 2002, pp. 15-23.

[4] Jonker, C.M. and Treur, J., Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

[5] Jonker, C.M., and Treur, J., A Temporal-Interactivist Perspective on the Dynamics of Mental States. *Cognitive Systems Research Journal*, vol.4, 2003, pp.137-155.

[6] Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. *Cognitive Systems Research Journal*, vol. 4(3), 2003, pp. 191-210.

[7] Keijzer, F., Representation in Dynamical and Embodied Cognition. *Cognitive Systems Research Journal*, vol. 3, 2002, pp. 275-288.

[8] Kim, J., *Philosophy of Mind*. Westview Press, 1996.

[9] Pokorny, L.R. and Ramakrishnan, C.R., Modeling and Verification of Distributed Autonomous Agents using Logic Programming. *Proc. of the Second International Workshop on Declarative Agent Languages and Technologies, DALT’04*. Lecture Notes in Artificial Intelligence, Springer Verlag, 2005.

[10] Pozos Parra, P., Nayak, A., Demolombe, R., Theories of Intentions in the Framework of Situation Calculus. *Proc. of the Second International Workshop on Declarative Agent Languages and Technologies, DALT’04*. Lecture Notes in Artificial Intelligence, Springer Verlag, 2005.

[11] Reiter, R., Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, 2001.

[12] Sun, R., Symbol grounding: a new look at an old idea. *Philosophical Psychology*, Vol.13, No.2, 2000, pp.149-172.

[13] Sun, R., *Duality of the Mind*. Lawrence Erlbaum Associates, 2002.

[14] Vasconcelos, W.W., Norm Verification and Analysis of Electronic Institutions. *Proc. of the Second International Workshop on Declarative Agent Languages and Technologies, DALT’04*. Lecture Notes in Artificial Intelligence, Springer Verlag, 2005.

[15] Vardi, M.Y., Branching vs. Linear Time: Final Showdown. *Proceedings of TACAS 2001 - Tools and Algorithms for the Construction and Analysis of Systems*. Genova, Italy, April 2-6, 2001. Lecture Notes in Computer Science, Volume 2031. New York, NY: Springer-Verlag, 2001, pp. 1-22.

[16] Walton, C., Model Checking Agent Dialogues. *Proc. of the Second International Workshop on Declarative Agent Languages and Technologies, DALT’04*. Lecture Notes in Artificial Intelligence, Springer Verlag, 2005.

PART VI REPRESENTATIONAL CONTENT

CHAPTER 17

Representational Content and Neural Mechanisms of Conditioning

Part of this chapter will appear as Bosse, T., Jonker, C.M., and Treur, J. (2005). Simulation of Conditioning Mechanisms in Agents. In: Balsa, J., Moniz, L., and Reis, L.P. (eds.), *Proceedings of the Third Workshop on Multi-Agent Systems: Theory and Applications, MASTA'05*.

An abstract of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2003). Representational Content and Agent-Environment Interaction. In: Schmalhofer, F., Young, R.M., and Katz, G. (eds.), *Proceedings of the European Cognitive Science Conference, EuroCogSci'03*. Lawrence Erlbaum Associates, London, p. 375.

Representational Content and Neural Mechanisms for Conditioning

Tibor Bosse[1], Catholijn M. Jonker[1], and Jan Treur[1,2]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, jonker, treur}@cs.vu.nl
http://www.cs.vu.nl/~{tbosse, jonker, treur}
[2] Universiteit Utrecht, Department of Philosophy,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Abstract. Conditioning is a form of learning that has been well-investigated. During such a learning process, by certain mechanisms internal states are created, based on (training) histories that extend over time. Usually such mechanisms and the states they create are described in non-representational physiological or algorithmic manners, for example in terms of living or artificial neural networks. Nevertheless, for the created internal states the question can be posed what is their representational content. The classical correlational approach to representational content requires a one-to-one correspondence between an internal state property of an agent and an external world state property. For internal mechanisms underlying conditioning this approach is not applicable, since internal states are based on training histories that extend over time. In this paper, alternative notions of representational content are explored with respect to their applicability for conditioning. Formal specifications of the representational content of internal state properties created by conditioning have been validated by automatically checking them on traces generated by a simulation model. The approach is illustrated for the case of Aplysia.

1. Introduction

Learning behaviour such as occurs in conditioning [9] can be analysed from two perspectives: the perspective of *externally* observable behaviour and the perspective of *internal* mechanisms to realise the behaviour. From the external perspective the dynamics of the observed behaviour can be analysed, i.e., how during a history of learning experiences the behaviour is changing. From an internal perspective the dynamics of the actual underlying neural mechanisms that play a role can be investigated and the behaviour they generate determined. One of the contributions of this paper is to relate the dynamics of models for these (internal) neural mechanisms to the dynamics of the externally observable behaviour.

Alternatively, models of other possible internal mechanisms can be designed and analysed. Such internal models have been developed from different traditions, varying from symbolic to connectionist and dynamic systems models. According to [3], behaviour can be described from three different perspectives:

1) biochemical
2) physiological/neural
3) behavioural

In general, models that belong to category 2) and 3) tend to get extremely complex and therefore not easy to handle. However, this paper introduces a high-level modelling

approach in which a neural description of conditioning still yields a manageable model. If the actual underlying neural mechanisms are taken as a point of departure to analyse conditioning, the sea hare *Aplysia* is an appropriate species to study, since its neural mechanisms have been well-investigated; cf. [2]. In this paper, as a second contribution, it will be shown how our modelling approach can be used to simulate *Aplysia's* neural mechanisms underlying conditioning.

For models that aim at the neural level, at forehand the notion of *representation* is not straightforward. Moreover, as mental states that result from conditioning processes have no simple one-to-one correspondence to states of the external world, especially in such cases the notion of representation is not unproblematic.

Classical approaches address representational content by correlations between an agent's internal state properties and external state properties; cf. [8], pp. 191-193. For example, the presence of a horse in the field is correlated to an internal state property that plays the role of a percept for this horse. A limitation is that internal state properties are to be related to (single) external states, and cannot be related to processes involving multiple states or events at different points in time. This shows that especially in cases where the agent learns from a number of trials extending over time, a classical approach to representational content is insufficient. Some authors even claim that it is a bad idea to aim for a notion of representation in such cases; e.g., [7], [10].

Kim [8], pp. 200-202, advocates as an alternative the *relational specification* approach where an internal state property is related to other states, distant in time and space. This approach allows to relate internal state properties to a number of other states at different points in time. Bickhard's *interactivist* approach [1] is another perspective that relates different states over time. In [5] it is shown how a temporal-interactivist approach to representational content of an internal state property can be formalized. Moreover, combining the temporal-interactivist perspective with the notion of relational specification, it is shown how to formalise representational content.

In this paper, as a third contribution, it is analysed how non-classical approaches may be used to define representational content in the case of conditioning processes and the internal states they create. In particular, for a case study it will be discussed how the temporal-interactivist approach and relational specification approach to representational content can be used. Moreover, it is shown how to formalise the criteria for representation in terms of dynamic properties that can be formally checked for given (e.g., simulated) traces of the agent-environment interaction.

An overview of the paper is as follows. In Section 2 the high-level modelling approach is briefly introduced. Section 3 introduces the case study and the state properties for this case study. In Section 4 the executable local dynamic properties describing basic mechanisms for the case study are presented; simulations on the basis of these local dynamic properties are discussed in Section 5. In Section 6 the interlevel relations between dynamic properties of the externally observable behaviour and the local properties describing the internal mechanisms are discussed. In Section 7 different approaches to representational content are explored and formalised. Section 8 discusses how all these dynamic properties have been checked against the simulation traces. Section 9 is a discussion.

2. Modelling Approach

To formally specify dynamic properties that express criteria for representational content from a temporal perspective an expressive language is needed. Dynamics will be described in the next section as evolution of *states* over time. The notion of state as used

here is characterised on the basis of an ontology defining a set of state properties that do or do not hold at a certain point in time. Dynamic properties can be formulated that relate a state at one point in time to one or more states at other points in time. A simple example is the following dynamic property specification:
'at any point in time t1 if the agent observes rain at t1, then there exists a point in time t2 after t1 such that at t2 the agent has internal state property s'

Here, for example, s can be viewed as a sensory representation of the rain. To express such dynamic properties, and other, more sophisticated ones, the temporal trace language TTL is used. Within this language, explicit references can be made to time points and traces. Here *trace or trajectory* over an ontology Ont is a time-indexed sequence of states over Ont. The sorted predicate logic temporal trace language TTL is built on atoms referring to, e.g., traces, time and state properties. For example, 'in the internal state of agent A in trace γ at time t property s holds' is formalised by state(γ, t, internal(A)) |= s. Here |= is a predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus. Dynamic properties are expressed by temporal statements built using the usual logical connectives and quantification (for example, over traces, time and state properties).

To be able to perform some (pseudo-)experiments, a simpler temporal language has been used to specify simulation models in a declarative manner. This language (the *leads to* language) enables to model direct temporal dependencies between two state properties in successive states. This executable format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the *leads to* language the notation $\alpha \rightarrow\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a precise definition of the leads to format in terms of the language TTL, see [6]. A specification of dynamic properties in leads to format has as advantages that it is executable and that it can often easily be depicted graphically. The *leads to* format has shown its value especially when temporal or causal relations in the (continuous) physical world are modelled and simulated in an abstract, non-discrete manner; for example, the intracellular chemistry of *E. coli* [4].

3. The Aplysia Case Study

In this Section the *Aplysia* case study will be introduced and the internal state properties and their dynamics to model this example are presented.

3.1. External Perspective

Aplysia is a sea hare that is often used to do experiments. It is able to learn; for example, it performs classical conditioning in the following manner. This (a bit simplified) description is mainly based on [2], pp. 155-156. First the (learning) behaviour viewed from an external perspective is addressed.

Behaviour before learning phase
Initially the following behaviour is shown:

- a tail shock leads to a response (contraction)
- a light touch on its siphon is insufficient to trigger such a response

Learning phase
Now suppose the following experimental protocol is undertaken. In each trial the subject is touched lightly on its siphon and then, shocked on its tail (as a consequence it responds).

Behaviour after a learning phase
It turns out that after a number of trials (three in the current example) the behaviour has changed:

- the animal also responds (contracts) on a siphon touch.

Note to characterise behaviour there is a difference between the *learned* behaviour (which is simply an *adapted* stimulus-response behaviour) and the *learning* behaviour, which is a form of *adaptive* behaviour, no stimulus-response behaviour. To specify such behaviours the following sensor and effector states are used: tail_shock, siphon_touch, contraction. In terms of these state properties the following global dynamic properties can be specified in *leads to* format:

GP1	tail_shock	$\twoheadrightarrow_{e,f,g,h}$	contraction	(always)
GP2	siphon_touch	$\twoheadrightarrow_{e,f,g,h}$	contraction	(after learning)

However the learning behaviour itself is not expressable in *leads to* format, but it is in TTL format:

GP3
at any point in time t,
if a siphon touch occurs
and at three different earlier time points t1, t2, t3,
a siphon touch occurred, directly followed by a tail shock
then it will contract

Formally:

$\forall\gamma\ \forall t\ \text{state}(\gamma, t) \models \text{siphon\_touch}\ \&$
$\exists t1, t2, t3, u1, u2, u3\quad t1 < u1 < t2 < u2 < t3 < u3 < t\ \&$
$\text{state}(\gamma, t1) \models \text{siphon\_touch}\ \&\ \text{state}(\gamma, u1) \models \text{tail\_shock}\ \&$
$\text{state}(\gamma, t2) \models \text{siphon\_touch}\ \&\ \text{state}(\gamma, u2) \models \text{tail\_shock}\ \&$
$\text{state}(\gamma, t3) \models \text{siphon\_touch}\ \&\ \text{state}(\gamma, u3) \models \text{tail\_shock}$
$\Rightarrow\ \exists t' \geq t\ \ \text{state}(\gamma, t') \models \text{contraction}$

As can be seen, the temporal complexity of the learning behaviour specification is much higher than that of the learned behaviour.

3.2. Internal Perspective

Roughly spoken the internal neural mechanism for Aplysia's conditioning can be depicted as in Figure 1; cf. [2]. A tail shock activates a sensory neuron SN1. Activation of this neuron SN1 activates the motoneuron MN; activation of MN makes the sea hare move. A siphon touch activates the sensory neuron SN2. Activation of this sensory neuron SN2 normally does not have sufficient impact on MN to activate MN. After learning, activation of SN2 has sufficient impact to activate MN. In addition, activation of SN1 also leads to activation of the intermediary neuron IN. If both SN2 and IN are activated simultaneously, this changes the synapse between SN2 and MN: it makes that in this synapse more neurotransmitter is produced if SN2 is activated. After a number of times this leads to the situation that also activation of SN2 leads to activation of MN.

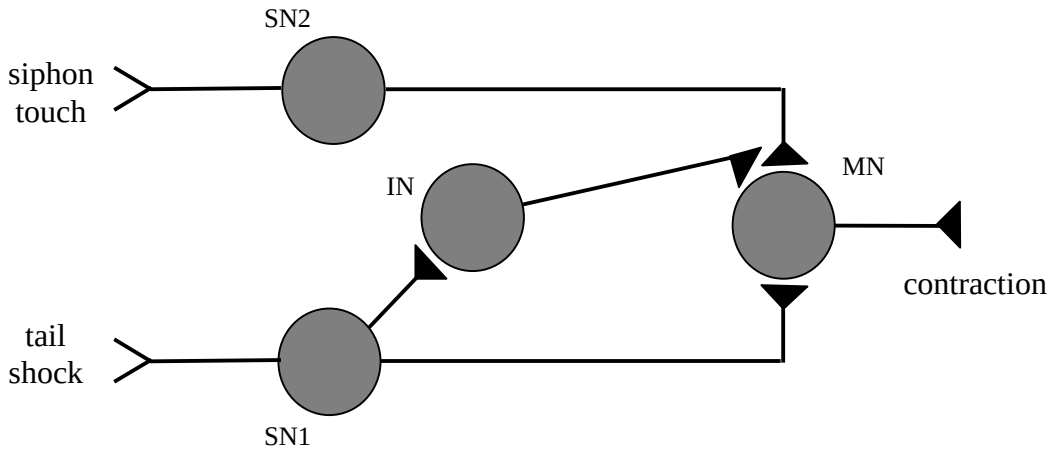


Figure 1. Neural mechanisms

To model the example the following internal state properties are used:

SN1	sensory neuron 1 is activated
SN2	sensory neuron 2 is activated
IN	intermediary neuron IN is activated
MN	motoneuron MN is activated
S(r)	the synapse between SN2 and MN is able to produce an amount r of neurotransmitter

The dynamics of these internal state properties involve temporal *leads to* relationships, which are analysed in more detail in the next section.

4. Local Dynamic Properties

To model the dynamics of the example, the following local properties (in *leads to* format) are considered. They describe the basic parts of the process. See also Figure 2 for a graphical overview of these local properties.

LP1	tail_shock	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	SN1
LP2	siphon_touch	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	SN2
LP3	SN1	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	IN $\wedge$ MN
LP4	S(r) $\wedge$ SN2 $\wedge$ IN $\wedge$ r < 4	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	S(r+1)
LP5	S(4) $\wedge$ SN2	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	MN
LP6	MN	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	contraction
LP7	S(r) $\wedge$ not S(r+1) $\wedge$ r < 4	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	S(r)
LP8	S(4)	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	S(4)
LP9	start	$\rightarrow\!\!\!\!\rightarrow_{e,f,g,h}$	S(1)

In Figure 2 an overview of these properties is given in a graphical form.

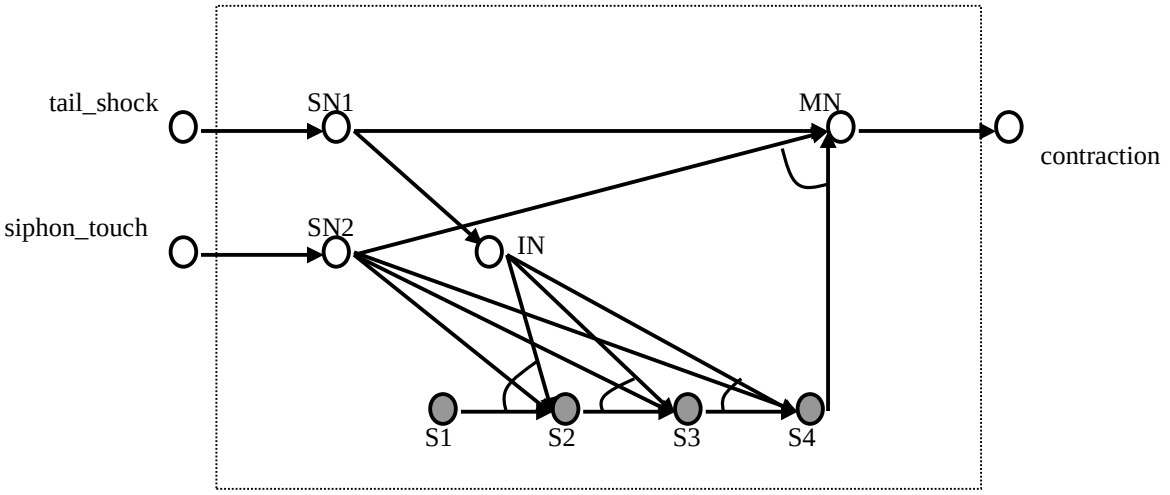


Figure 2. Overview of the basic dynamics of the simulation model

5. Simulation

A special software environment has been created to enable the simulation of executable models. Based on an input consisting of dynamic properties in *leads to* format, the software environment generates simulation traces. An example of such a trace can be seen in Figure 3. Here, time is on the horizontal axis, the state properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false. This trace is based on all local properties identified above. In property LP1 and LP2 the values (0,0,1,3) have been chosen for the timing parameters e, f, g, and h. In all other properties, the values (0,0,1,1) have been chosen. As can be seen in Figure 3, at the beginning of the trace the organism has not performed any conditioning. The initial siphon touch it receives does lead to the activation of sensory neuron SN2, but the synapse between SN2 and motoneuron MN does not produce much neurotransmitter yet (indicated by internal state property S(1)). Thus, the activation of SN2 does not yield an activation of MN, and consequently no external action follows. In contrast, it is shown that a shock of the organism's tail does initially lead to the external action of contraction. This can be seen in Figure 3 between time point 10 (when the tail shock occurs) and time point 13 (when the animal contracts). After that, the actual learning phase starts. This phase consists of a sequence of three trials where a siphon touch is immediately followed by a tail shock. As a result, the sensory neuron SN2 is activated at the same time as the intermediary neuron IN, which causes the synapse to change so that it can produce an increased amount of neurotransmitter each time SN2 is activated. Such a change in the synapse is indicated by a transition from one internal state property to another (first from S(1) to S(2), then to S(3), and finally to S(4)). As soon as internal state property S(4) holds (see time point 44), the conditioning process has been performed successfully. From that moment, Aplysia's behaviour has changed: it also contracts on a siphon touch.

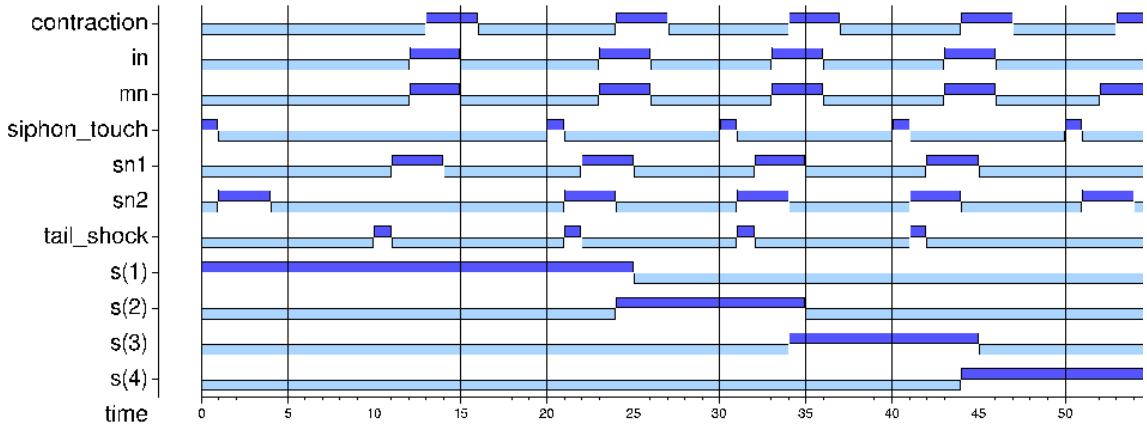


Figure 3. Example simulation trace

As a side remark, notice that the amount of trials (three) is kept low to keep the example simple. However, similar experiments have been performed with a case of 1000 learning steps. Since the abstract way of modelling used for the simulation is not computationally expensive, also these simulations took no more than 90 seconds. In addition, our simulation approach has possibilities to incorporate real numbers in state properties, and to perform complex mathematical operations with these numbers. This makes it more expressive than more traditional forms of temporal logic.

6. Interlevel Relations

This section outlines the interlevel connections between dynamic properties at different levels, varying from dynamic properties at the local level of basic parts of the process to dynamic properties at the global level of the overall process. The following interlevel relations between local dynamic properties and nonlocal dynamic properties can be identified. The local properties together imply the global property GP3 that experiencing the combination of a tail shock and a siphon touch three times, results in a response to the siphon touch alone. The exact relationship between the local properties and GP3 requires one additional property, i.e., CWA:

CWA (Closed World Assumption)
The second-order property that is commonly known as the Closed World Assumption expresses that at any point in time a state property that is not implied by a specification to be true is false. Let Th be the set of all local properties LP1 through LP9, then the formalisation is:
$\forall P \in \mathrm{At(ONT)}\ \forall \gamma\ \forall t{:}\mathrm{Th}\ |\!\!\sim\!\!/\ \mathrm{state}(\gamma, t) \models P \Rightarrow \mathrm{state}(\gamma, t) \models \mathrm{not}\ P$

The Closed World Assumption is needed to ensure that the intermediate results as indicated by the S(r) state properties can only hold as a result of the local properties LP1 through LP9, and not because of some other (mysterious) cause. The relationship between the local properties, CWA, and GP3 is:

(1) LP1 through LP9 & CWA $\Rightarrow$ GP3

Essential milestones in the proof of relationship (1) are that subsequently S(1), S(2), S(3), and (S4) will hold. These milestones can be seen as the result of a learning process. Therefore, an additional lemma is introduced. This lemma describes the effect of a learning step on the synapse, showing the increase of parameter r in state property S(r) given that the siphon is touched within a tiny interval before the tail is shocked. In this

case study the effect we are interested in is already reached at r=4. The lemma can easily be adapted for more lengthy learning processes. Formally, the lemma is specified as:

M(g, h, r) Learning step
$\forall\gamma\ \forall t1, t2, u1$
$t1 < u1 < t1 + g\ \&\ t1 < t2 < t1 + g\ \&\ r < 4\ \&$
$\forall t\ [t1 \le t < t1 + h \Rightarrow state(\gamma, t) \models siphon\_touch\ ]\ \&$
$\forall t\ [u1 \le t < u1 + h \Rightarrow state(\gamma, t) \models tail\_shock\ ]\ \&$
$\forall t\ [t2 \le t < t2 + h \Rightarrow state(\gamma, t) \models S(r)\ ]$
$\Rightarrow$
$\exists t3\ [\ t3 \ge t2\ \&\ \forall t\ [t3 \le t < t3 + h \Rightarrow state(\gamma, t) \models S(r+1)\ ]\ ]$

Property M(g,h,r) can be proved for g=1, h=1, and r varying from 1 to 4 from LP1, LP2, LP3, LP4, LP7, and CWA, taking (0, 0, 1, 3) as timing parameters in LP1 and LP2, and (0, 0, 1, 1) for the timing parameters of the other local properties.

(2) $\quad LP1\ \&\ LP2\ \&\ LP3\ \&\ LP4\ \&\ LP7\ \&\ CWA \Rightarrow M(1, 1, r)$

The full proof is left out of this paper. Instead only a sketch of the proof is given, in which some initialisation issues are ignored. The crucial points are that the siphon touch and the tail shock are coordinated in time such that SN2 (by application of LP2) exists long enough for it to co-exist with IN. Given LP7 and CWA it becomes clear that S(r) persists long enough for LP4 to have effect. The following sketch is illustrated by Figure 4.

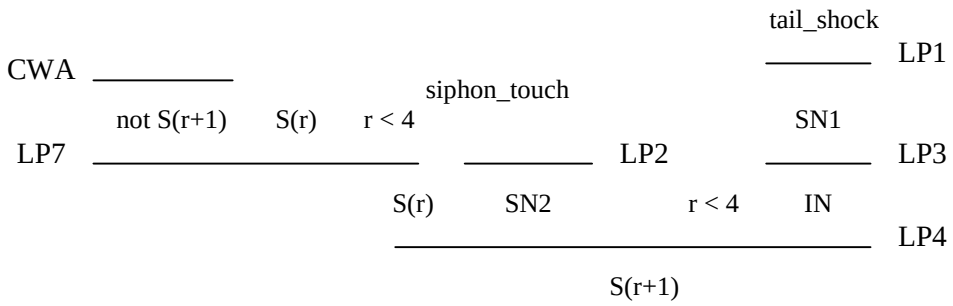


Figure 4. Sketch of interlevel relationship (2)

Assume that

$t1 < u1 < t1 + g\ \&\ t1 < t2 < t1 + g\ \&\ r < 4\ \&$
$\forall t\ [t1 \le t < t1 + 1 \Rightarrow state(\gamma, t) \models siphon\_touch\ ]\ \&$
$\forall t\ [u1 \le t < u1 + 1 \Rightarrow state(\gamma, t) \models tail\_shock\ ]\ \&$
$\forall t\ [t2 \le t < t2 + 1 \Rightarrow state(\gamma, t) \models S(r)\ ]$

Then CWA can be applied to derive that

$\forall t\ [t \le t2 \Rightarrow state(\gamma, t) \models not\ S(r+1)\ ]$

In addition LP1 can be applied to derive

$\forall t\ [u1 + 1 \le t < u1 + 4 \Rightarrow state(\gamma, t) \models SN1\ ]$

Using this information, and LP3, the following is derived

$\forall t\ [u1 + 2 \le t < u1 + 5 \Rightarrow state(\gamma, t) \models IN\ ]$

Similarly, LP2 can be applied on the time duration of the siphon touch to derive:

$\forall t \; [t1 + 1 \leq t < t1 + 4 \Rightarrow state(\gamma, t) \models SN2]$

In order to apply LP4 on the intersection interval of the periods during which both SN2 and IN hold (i.e., $[u1 + 2, t1 + 4>$, which has a duration > 1), it must be ascertained that S(r) also holds long enough (at least 1) in that interval. Since $t1 < u1 < t1 + g$ & $t1 < t2 < t1 + g$, the absolute difference $|t2 – u1| < 1$.

For the other case LP7 needs to applicable to derive a persistence of S(r), to the extent that it overlaps long enough SN2 and IN. Given that LP4 cannot be applied in the time period $[t2, t2 + 1>$, and the fact that there is no local property that derives S(r+1) during that same interval, CWA is applicable. Even stronger, CWA is applicable until LP4 is applicable, deriving:

$\forall t \; [t2 \leq t < u1 + 3 \Rightarrow state(\gamma, t) \models S(r)]$

Note that given that $t1 < u1 < t1 + g$ (with g=1), the interval $[t2, u1 + 3>$ overlaps with $[u1 + 2, t1 + 4>$ for the interval $[u1 + 2, u1 + 3>$. Therefore, LP4 can be applied on this interval to derive the result of M(1,1,r), thus proving relationship (2):

$\forall t \; [u1 + 3 \leq t < u1 + 4 \Rightarrow state(\gamma, t) \models S(r+1)]$

As can be seen from the proof sketch, the timing issues make proofs complex. In Figure 4, the timing information is left out. The tree only gives insight into which local properties (or CWA) are applied on which state properties.

The proof sketch for interlevel relationship (1) (as illustrated in Figure 5, again with all timing elements left out) takes all the siphon touches, tail shocks as given in the precondition of the implication in GP3 as hypotheses and shows how to derive a contraction of the Aplysia.

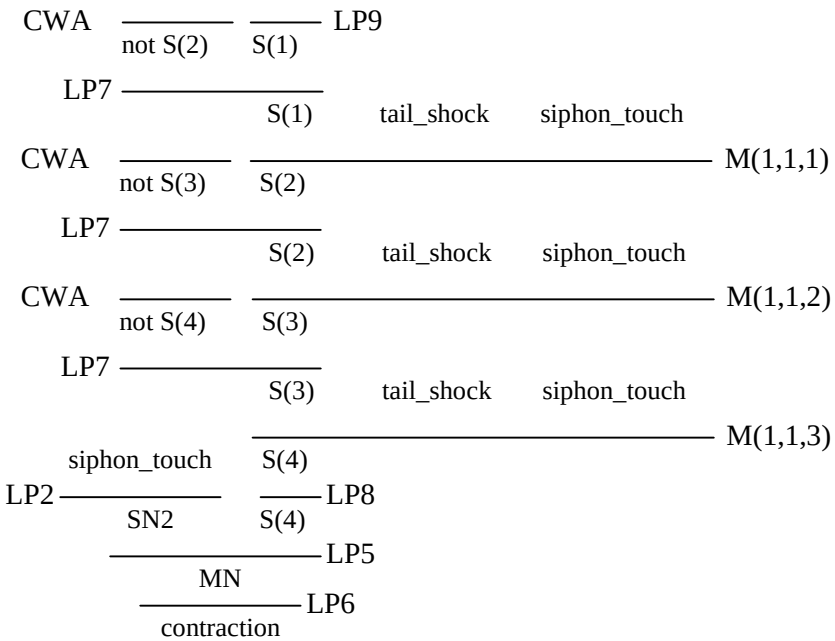


Figure 5. Sketch of interlevel relationship (1)

The initial assumption LP9 provides that S(1) holds in the beginning. By the CWA it is possible to apply LP7 ensuring that S(1) persists until the first sequence of siphon touch and tail shock have taken place and M(1,1,1) can be applied. The pattern of applying CWA and LP7 to ensure persistence is repeated for every new occurrence of the sequence of siphon touch and tail shock, until S(4) holds. Because of LP8, S(4) persists until a new siphon touch occurs, and LP2 has been applied leading to SN2. The persisting of S(4) and the existence of SN2 make LP5 applicable, leading to MN. Finally, the application of LP6 on MN leads to a contraction, thus completing the proof of (1).

7. Representational Content

In the literature on Philosophy of Mind different types of approaches to representational content of an internal state property have been put forward, for example the causal/correlational, interactivist and relational specification approach; cf. [1], [8], pp. 191-193, 200-202. These approaches to representational content have in common that the occurrence of the internal state property at a specific point in time is related to the occurrence of other state properties, at the same or at different time points. The 'other state properties' can be of two types:

A. external world state properties, independent of the agent

B. the agent's sensor state and effector state properties, i.e. the agent's interaction state properties (interactivist approach)

Furthermore, the type of relationships can be (1) purely functional *one-to-one correspondences*, (e.g., the correlational approach), or (2) they can involve more *complex relationships* with a number of states at different points in time in the past or future, (e.g., the interactivist approach). So, four types of approaches to representational contents are distinguished, that can be indicated by codings such as A1, A2, and so on. Below, examples of such approaches are given.

According to the causal/correlational approach, the representational content of a certain internal state is given by a one-to-one correlation to another (in principle external) state property: type A1. Such an external state property may exist backward as well as forward in time. Hence, for the current example, in order to define the representational content of an internal state property, one should try if this can be related to a world state property that either existed in the past or will exist in the future. For example, the representational content for internal state property SN1 can be defined as world state property tail_shock, by looking backward in time. However, for some of the other internal state properties the representational content cannot be defined adequately according to the causal/correlational approach. In these cases, reference should not be made to one single state in the past or in the future, but to a temporal sequence of inputs or output state properties, which is not considered to adequately fit in the correlational approach.

The Temporal-Interactivist approach [1], [5] relates the occurrence of internal state properties to sets of past and future interaction traces: type B. In this paper the focus is on the B2 type, which is the more advanced case.

The Relational Specification approach to representational content is based on a specification of how the occurrence of an internal state property relates to properties of states distant in space and time; cf. [8], pp. 200-202. In this paper it is used in conjunction with the temporal-interactivist approach. Thus, the representational content of a certain internal state can be defined by specifying a temporal relation of the internal state property to sensor and action states in the past and future. An overview for the content of all internal state properties according to the temporal relational specification approach is

given, in an informal notation, in Table 1. Note that these relationships are defined at a semantic level. Different interaction state properties, separated by commas, should be read as the temporal sequence of these states.

Internal State Property	Content (backward)	Content (forward)
S(2)	siphon_touch, tail_shock	
S(3)	siphon_touch, tail_shock, siphon_touch, tail_shock	
S(4)	siphon_touch, tail_shock, siphon_touch, tail_shock, siphon_touch, tail_shock	any siphon_touch is followed by contraction

Table 1. Temporal-Interactivist Approach (semantic level)

Table 2 and 3 describe the same information as Table 1, but this time syntactically, expressed by TTL formulae. The following abstractions are used to describe training periods:

training_up_to(γ, t1, u1, 1) ≡ u1 = t1 + 1 &
state(γ, t1) ⊨ siphon_touch & state(γ, u1) ⊨ tail_shock
training_up_to(γ, t1, u2, 2) ≡
∃u1, t2 [u1 < t2 & u2 = t2 + 1]
training_up_to(γ, t1, u1, 1) &
state(γ, t2) ⊨ siphon_touch & state(γ, u2) ⊨ tail_shock
training_up_to(γ, t1, u3, 3) ≡
∃u2, t3 [u2 < t3 & u3 = t3 + 1]
training_up_to(γ, t1, u2, 2) &
state(γ, t3) ⊨ siphon_touch & state(γ, u3) ⊨ tail_shock

I.s.p.	Content (backward)
S(2)	∀t1, u1 [training_up_to(γ, t1, u1, 1) & ¬∃t0 [training_up_to(γ, t0, u1, 2)] ⇒ ∃t2 > u1 [state(γ, t2) ⊨ S(2)]] ∀t1, u2 [training_up_to(γ, t1, u2, 2) & ¬∃t0 [training_up_to(γ, t0, u2, 3)] ⇒ ∃t3 > u2 [state(γ, t3) |=/= S(2)]]
S(3)	∀t1, u2 [training_up_to(γ, t1, u2, 2) & ¬∃t0 [training_up_to(γ, t0, u2, 3)] ⇒ ∃t3 > u1 [state(γ, t3) ⊨ S(3)]] ∀t1, u3 [training_up_to(γ, t1, u3, 3) & ¬∃t0 [training_up_to(γ, t0, u3, 4)] ⇒ ∃t4 > u3 [state(γ, t4) |=/= S(3)]]
S(4)	∀t1, u3 [training_up_to(γ, t1, u3, 3) & ¬∃t0 [training_up_to(γ, t0, u3, 4)] ⇒ ∃t4 > u3 [state(γ, t4) ⊨ S(4)]]

Table 2. Temporal-Interactivist Approach (syntactic level, backward)

I.s.p.	Content (forward)
S(4)	∃ t' ≥ t [state(γ, t') ⊨ siphon_touch & ∀t' ≥ t [state(γ, t') ⊨ siphon_touch ⇒ ∃t" ≥ t' state(γ, t") ⊨ contraction]

Table 3. Temporal-Interactivist Approach (syntactic level, forward)

Consider, for example, the backward representational content of state property S(2). According to Table 2, the occurrence of exactly one learning trial (indicated by the fact that at u1, a training period up to 1 but not up to 2 has passed) eventually leads to a time

point where S(2) holds. In addition, to make the content more precise, it is specified that the occurrence of exactly two learning trials eventually causes S(2) not to hold.

8. Checking Dynamic Properties

In addition to the simulation software, a software environment has been developed that enables to check dynamic properties specified in TTL against simulation traces. This software environment takes a dynamic property and one or more (empirical or simulated) traces as input, and checks whether the dynamic property holds for the traces. Traces are represented by sets of Prolog facts of the form

```
holds(state(m1, t(2)), a, true).
```

where m1 is the trace name, t(2) time point 2, and a is a state formula in the ontology of the component's input. It is indicated that state formula a is true in the component's input state at time point t2. The programme for temporal formula checking basically uses Prolog rules for the predicate sat that reduce the satisfaction of the temporal formula finally to the satisfaction of atomic state formulae at certain time points, which can be read from the trace representation. Examples of such reduction rules are:

```
sat(and(F,G)) :- sat(F), sat(G).
sat(not(and(F,G))) :- sat(or(not(F), not(G))).
sat(or(F,G)) :- sat(F).
sat(or(F,G)) :- sat(G).
sat(not(or(F,G))) :- sat(and(not(F), not(G))).
```

Using automatic checks of this kind, many of the properties presented in this paper have been checked against traces such as the one depicted in Figure 3. In particular, dynamic property GP3 (expressing the learning behaviour) has been checked successfully. Furthermore, the properties for representational content denoted in Table 2 have been checked. The duration of these checks varied from 1 to 3 seconds, depending on the complexity of the formula. They all turned out to be successful, which validates (for the given traces at least) our choice for the representational content of the internal state properties. However, note that these checks are only an empirical validation, they are no exhaustive proof as, e.g., model checking is. Currently, the possibilities are explored to combine TTL with existing model checking techniques.

9. Discussion

This paper contributes an analysis of the dynamics of classical conditioning from a logical perspective. It provides two types of temporal logical formalisations, one at the behavioural level, and one at the neurological level; cf. [3]. The neural processes of the *Aplysia* case study (cf. [2]) have been formalised by identifying executable local dynamic properties for the basic dynamics of *Aplysia*'s neural conditioning mechanism. On the basis of these local properties simulations have been made. Moreover, it is shown how the descriptions at these two levels (i.e., the level of the neurological mechanisms and of the overall behaviour) can be logically related to each other, which can be considered as a formalisation of the (inter-level) reduction relations between the two levels.

Usually mechanisms for conditioning and the states they create are described in non-representational physiological or algorithmic manners, for example in terms of living or artificial neural networks. Nevertheless, for the created internal states the question can be

posed what is their representational content. The classical causal/correlational approach to representational content (cf. [8], pp. 191-193) requires a one-to-one correspondence between an internal state property of an agent and an external world state property. For adaptive agents that learn from experiences, this classical approach does not suffice. In particular, an internal state in such an agent does not depend on just one state property of the external world, but is affected by a history of events.

In this paper, as a second challenge, it was also explored how representational content can be defined for conditioning using approaches such as in [1], [5], [8], pp. 200-202, and, especially, how it can be formalised. The specifications of the representational content of the internal (neural) state properties for *Aplysia* have been validated by automatically checking them on the traces generated by the simulation model. This shows that the internal (neural) state properties indeed fulfil the representational content specification.

References

[1] Bickhard, M.H., Representational Content in Humans and Machines. *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 5, 1993, pp. 285-333.

[2] Gleitman, H., *Psychology*, W.W. Norton & Company, New York, 1999.

[3] Hawkins, R.D., and Kandel, E.R. (1984). Is There a Cell-Biological Alphabet for Simple Forms of Learning? *Psychological Review*, vol. 91, pp. 375-391.

[4] Jonker, C.M., Snoep, J.L., Treur, J., Westerhoff, H.V., and Wijngaards, W.C.A.. BDI-Modelling of Intracellular Dynamics. In: A.B. Williams and K. Decker (eds.), *Proc. of the First International Workshop on Bioinformatics and Multi-Agent Systems, BIXMAS'02*, 2002, pp. 15-23.

[5] Jonker, C.M., and Treur, J., A Temporal-Interactivist Perspective on the Dynamics of Mental States. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 137-155.

[6] Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 191-210.

[7] Keijzer, F. Representation in Dynamical and Embodied Cognition. *Cognitive Systems Research Journal*, vol. 3, 2002, pp. 275-288.

[8] Kim, J., *Philosophy of Mind*. Westview Press, 1996.

[9] Pavlov, I.P. *Conditioned reflexes*. Oxford: Oxford University Press, 1927.

[10] Sun, R. Symbol grounding: a new look at an old idea. *Philosophical Psychology*, Vol.13, No.2, 2000, pp.149-172.

PART VI REPRESENTATIONAL CONTENT

CHAPTER 18

Simulation and Representation of Body, Emotion, and Core Consciousness

This chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2005). Simulation and Representation of Body, Emotion, and Core Consciousness. In: *Proceedings of the AISB 2005 Symposium on Next Generation approaches to Machine Consciousness: Imagination, Development, Intersubjectivity, and Embodiment*, pp. 95-103.

Simulation and Representation of Body, Emotion, and Core Consciousness

Tibor Bosse[1], Catholijn M. Jonker[2], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, treur}@cs.vu.nl
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl

Abstract. This paper contributes an analysis and formalisation of Damasio's theory on core consciousness. Three important concepts in this theory are "emotion", "feeling", and "feeling a feeling" (or core consciousness). In particular, a simulation model is described of the neural dynamics leading via emotion and feeling to core consciousness, and dynamic properties are formally specified that hold for these dynamics. These properties have been automatically checked for the simulation traces. Moreover, a formal analysis is made and verified of relevant notions of representation.

1. Introduction

In (Damasio, 2000) the neurologist Antonio Damasio puts forward his theory of consciousness. He describes his theory in an informal manner, and supports it by a vast amount of evidence from neurological practice. More experimental work supporting his theory is reported in (Damasio et al., 2000; Parvizi and Damasio, 2001). Damasio's theory is described on the one hand in terms of the occurrence of certain neural states (or neural patterns), and temporal or causal relationships between them. Formalisation of these relationships requires a modelling format that is able to express direct temporal or causal dependencies. On the other hand Damasio gives interpretations of most of these neural states as representations, for example as 'sensory representation', or 'second-order representation'. This requires an analysis of what it means that a neural state is a representation for something. This paper focuses on Damasio's notions of 'emotion', 'feeling', and 'core consciousness' or 'feeling a feeling'. In (Damasio, 2000), Damasio describes an *emotion as neural object* (or *internal emotional state*) as an (unconscious) neural reaction to a certain stimulus, realized by a complex ensemble of neural activations in the brain. As the neural activations involved often are preparations for (body) actions, as a consequence of an internal emotional state, the body will be modified into an *externally observable emotional state*. Next, a *feeling* is described as the (still unconscious) sensing of this body state. Finally, *core consciousness* or *feeling* a *feeling* is what emerges when the organism detects that its representation of its own body state (the *proto-self*) has been changed by the occurrence of the stimulus: it becomes (consciously) aware of the feeling.

This paper aims at formalisations and simulation models for these three notions. In addition, the notion of representation used by Damasio is formally analysed against different approaches to representational content from the literature on the Philosophy of Mind. It is shown that the classical causal/correlational approach to representational content, e.g., (Kim, 1996), pp. 191-193, is inappropriate to describe the notion of

representation for core consciousness used by Damasio, as this notion essentially involves more complex temporal relationships describing histories of the organism's interaction with the world. An alternative approach is shown to be better suited: representational content as relational specification over time and space, cf. (Kim, 1996), pp. 200-202. Criteria for this approach are formalised, and it is shown that the formalisation of Damasio's notions indeed fit these criteria.

A brief summary of the main basic assumptions underlying Damasio's approach is expressed in:

> *'First, I am suggesting that (…) 'having a feeling' is not the same as 'knowing a feeling', that reflection on feeling is yet another step up. (…) The inescapable and remarkable fact about these three phenomena – emotion, feeling, consciousness – is their body relatedness. (…) As the representations of the body grow in complexity and coordination, they come to constitute an integrated representation of the organism, a proto-self. Once that happens, it becomes possible to engender representations of the proto-self as it is affected by interactions with a given environment. It is only then that consciousness begins, only thereafter that an organism that is responding beautifully to its environment begins to discover that* it *is responding beautifully to its environment. But all of these processes – emotion, feeling, and consciousness – depend for their execution on representations of the organism. Their shared essence is the body. (Damasio, 2000), pp. 283-284.*

In Section 2 the modelling approach used is briefly introduced. In Sections 3, 4, and 5, for a simple example models are presented for the processes leading to emotion, feeling, and feeling a feeling (or conscious feeling), respectively. Section 6 provides the results of a simulation of these models. In Section 7 it is analysed in how far the representational content of Damasio's notions can be described by two approaches from Philosophy of Mind. Formalisations of some of the dynamic properties of the processes leading to emotion, feeling and feeling a feeling are presented. Next, Section 8 addresses verification. It is shown that the notions for representational content developed in Section 7 indeed hold for the model. The verification is performed both by automated checks and by mathematical proof. Section 9 concludes the paper with a discussion.

2. Modelling Approach

To model the making of emotion, feeling and core consciousness, dynamics play in important role. Dynamics will be described in the next section as evolution of *states* over time. The notion of state as used here is characterised on the basis of an ontology defining a set of state properties that do or do not hold at a certain point in time. The modelling perspective taken is not a symbolic perspective, but essentially addresses the neural processes and their dynamics as neurological processes. This implies that states are just neurological states. To successfully model such complex processes, forms of abstraction are required; for example:

- neural states or activation patterns are modelled as single state properties
- large-dimensional vectors of such (distributed) state properties are composed to one single composite state property, when appropriate; e.g., (p1, p2, …) to p and (S1, S2, …) to S in Section 3.

To describe the dynamics of the processes mentioned above, explicit reference is made to time. Dynamic properties can be formulated that relate a state at one point in time to a state at another point in time. A simple example is the following dynamic property specification for belief creation based on observation:

'at any point in time t1, if the agent observes rain at t1,
then there exists a point in time t2 after t1 such that at t2 the agent has internal state property s'

Here, for example, s can be viewed as a sensory representation of the rain. To express dynamic properties in a precise manner a language is used in which explicit references can be made to time points and traces: the Temporal Trace Language TTL; cf. (Jonker and Treur, 2002). Here a *trace or trajectory* over an ontology Ont is a time-indexed sequence of states over Ont. The sorted predicate logic temporal trace language TTL is built on atoms referring to, e.g., traces, time and state properties. For example, 'in the internal state of agent A in trace γ at time t property s holds' is formalised by state(γ, t, internal(A)) |= s. Here |= is a predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus. Dynamic properties are expressed by temporal statements built using the usual logical connectives and quantification (for example, over traces, time and state properties).

To be able to perform some (pseudo)-experiments, a simpler temporal language has been used to specify simulation models in a declarative manner. This language (the *leads to* language) enables to model direct temporal dependencies between two state properties in successive states. This executable format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h, non-negative real numbers. In the *leads to* language the notation $\alpha \rightarrow\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α hold for a time interval with duration g, then after some delay (between e and f) state property β will hold for a time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see (Jonker, Treur, and Wijngaards, 2003). A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

In Sections 3, 4, 5 and 6, the *leads to* format has been used to create simulation models of the processes leading to emotion, feeling and core consciousness in terms of neural processes. Given this physical-level model and its dynamic properties, a next step is to assign representational content to (some of) the relevant state properties. For nontrivial cases representational content involves histories of interaction between organism and world (Bickhard, 1993; Jonker and Treur, 2003), and this also shows up in Damasio's theory. To specify and analyse the representational content to a number of state properties of the models and the traces they generate, the more expressive TTL format is used in Section 7. Both formats are used in Section 8.

3. Emotion

First Damasio's notion of *emotion* is addressed. He explains this notion as follows:

> *'The substrate for the representation of emotions is a collection of neural dispositions in a number of brain regions (…) They exist, rather, as potential patterns of activity arising within neuron ensembles. Once these dispositions are activated, a number of consequences ensue. On the one hand, the pattern of activation represents, within the brain, a particular emotion as 'neural object'. On the other, the pattern generates explicit responses that modify both the state of the body proper and the state of other brain regions. By so doing, the responses create an emotional state, and at that point, an external observer can appreciate the emotional engagement of the organism being observed. (Damasio, 2000), p. 79.*

According to this description, an *internal emotional state* is a collection of neural dispositions in the brain, which are activated as a reaction on a certain stimulus. Once such an internal emotional state occurs, it entails modification of both the body state and the state of other brain regions. By these events, an *external emotional state* is created, which is accessible for external observation.

Assume that the music you hear is so special that it leads to an emotional state in which you show some body responses on it (e.g., shivers on your back). This process is described by executable local dynamic properties taking into account internal state properties sr(music) for activated sensory representation of hearing the music, and (p1, p2, …) a vector for the activation of preparatory states for the body responses (S1, S2, …); see Figure 1.

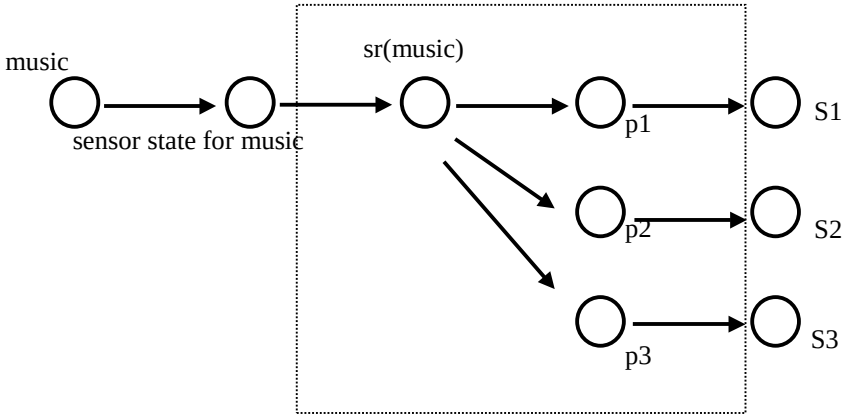


Figure 1. Processes leading to an emotional state

These vectors are the possible internal emotional states. Note that the state properties are abstract in the sense that a state property refers to a specific neural activation pattern. In the model the conjunction p1 & p2 & .. of these preparatory state properties is denoted by p; this p can be considered a composite state property. Moreover, the conjunction of the vector of all body state properties responding to the music S1, S2, … (i.e., the respective body state properties for which p1, p2, ... are preparing) is denoted by (composite) state property S.

The model abstracted in this manner is depicted in Figure 2, upper part. In formal textual format these local properties are as follows:

LP0 music •→» sensor_state(music)
LP1 sensor_state(music) •→» sr(music)
LP2 sr(music) •→» p
LP3 p •→» S

In the remainder of this paper this abstract type of modelling will be used. Notice, however, that each of the abstract state properties used are realised in the organism in a distributed manner as a large-dimensional vector of more local (neural) state properties. Also the sensory representation sr(music) may be considered such a composite state property with different aspects of the music represented in different forms at different places. Notice, moreover, that the names of the state properties have been chosen to support readability for humans. But in principle these names should be considered as neutral indications of neural states, such as n1, n2, and so on.

4. Feeling

Next, Damasio's notion of *feeling* is considered. He expresses the emergence of feeling as follows:

As for the internal state of the organism in which the emotion is taking place, it has available both the emotion as neural object (the activation pattern at the induction sites) and the sensing of the consequences of the activation, a feeling, provided the resulting collection of neural patterns becomes images in mind. (…) The changes related to body state are achieved by one of two mechanisms. One involves what I call the 'body loop'. (…) .. the body landscape is changed and is subsequently represented in somatosensory structures of the central nervous system, from the brain stem on up. The change in the representation of the body landscape can partly be achieved by another mechanism, which I call the 'as if body loop'. In this alternate mechanism, the representation of body-related changes is created directly in sensory body maps, under the control of other neural sites, for instance, the prefrontal cortices. It is 'as if' the body had really been changed but it was not. (…) Assuming that all the proper structures are in place, the processes reviewed above allow an organism to undergo an emotion, exhibit it, and image it, that is, feel the emotion. (Damasio, 2000), pp. 79-80.

Thus, a feeling emerges when the collection of neural patterns contributing to the emotion lead to mental images. In other words, the organism senses the consequences of the internal emotional state. Damasio distinguishes two mechanisms by which a feeling can be achieved:

1) Via the *body loop*, the internal emotional state leads to a changed state of the body, which subsequently, after sensing, is represented in somatosensory structures of the central nervous system.
2) Via the *as if body loop*, the state of the body is not changed. Instead, on the basis of the internal emotional state, a changed representation of the body is created directly in sensory body maps. Consequently, the organism experiences the same feeling as via the body loop: it is 'as if' the body had really been changed but it was not.

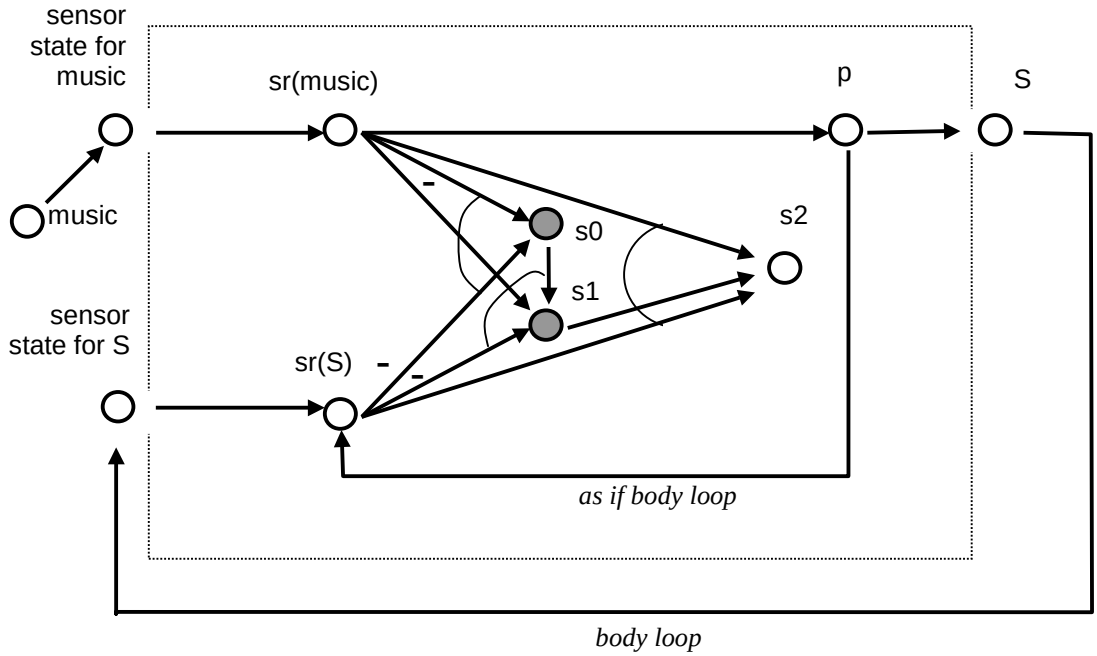


Figure 2. Overview of the simulation model

The model described in Section 3 can be extended to include a number of internal state properties for sensory representations of body state properties that are changed due to responses on the music; together these sensory representations constitute the feeling induced by the music. In Figure 2 the conjunction of these sensory representations is depicted: sr(S) (a sensory representation of the changed body state; this may be

materialised in a distributed manner as a kind of vector). This describes the 'body loop' for the responses on the music; here S and sensor_state(S) are effects and sensors in the body, respectively. In formal format, two additional local dynamic properties are needed (see also Figure 2):

LP4 S $\bullet\!\!\rightarrow\!\!\!\!\rightarrow$ sensor_state(S)
LP5 sensor_state(S) $\bullet\!\!\rightarrow\!\!\!\!\rightarrow$ sr(S)

Notice that an internal state property sr(shivering) for shivering only, does not directly relate to the music. It is caused by the external stimulus shivering, which in this particular case is originally caused by the music. This body state property shivering could be present for a lot of other reasons as well, e.g., a cold shower. However, taking into account that not only shivering but a larger number of sensory state properties constitute the overall composite state property sr(S), the feeling will be more unique for the music. For the case of an 'as if body loop' dynamic properties LP3, LP4 and LP5 can be replaced by the following local dynamic property directly connecting p and sr(S).

LP6 p $\bullet\!\!\rightarrow\!\!\!\!\rightarrow$ sr(S)

Also a combination of models can be made, in which some effects of hearing the music is caused by a body loop and some are caused by an 'as if body loop'.

5. Feeling a Feeling

Finally, Damasio's notion of *knowing* or *being conscious of* or *feeling* a *feeling* is addressed. This notion is based on the organism detecting that its representation of its own (body) state (the *proto-self*) has been changed by the occurrence of a certain object (the music in our example). According to Damasio, the proto-self is

> *"a coherent collection of neural patterns which map, moment by moment, the state of the physical structure of the organism". (Damasio, 2000), p. 177.*

He expresses the way in which the proto-self contributes to a conscious feeling in the following hypothesis:

> *Core consciousness occurs when the brain's representation devices generate an imaged, nonverbal account of how the organism's own state is affected by the organism's processing of an object, and when this process enhances the image of the causative object, thus placing it in a spatial and temporal context. (p. 169)… with the license of metaphor, one might say that the swift, second-order nonverbal account narrates a story:* that of the organism caught in the act of representing its own changed state as it goes about representing something else. *But the astonishing fact is that the knowable entity of the catcher has just been created in the narrative of the catching process. (…) You know it is* you *seeing because the story depicts a character – you – doing the seeing. (pp. 170-172) … beyond the many neural structures in which the causative object and the proto-self changes are separately represented, there is at least one other structure which* re-represents *both proto-self and object in their temporal relationship and thus represent what is actually happening to the organism:* proto-self at the inaugural instant; object coming into sensory representation; changing of inaugural proto-self into proto-self modified by object. *(p. 177).*

In summary, the conscious feeling occurs when the organism detects the transitions between the following moments:

1. The proto-self exists at the inaugural instant.
2. An object comes into sensory representation.
3. The proto-self has become modified by the object.

For our case we restrict ourselves to placing the relevant events in a temporal context. In a detailed account, in the trace considered subsequently the following events take place: no sensory representations for music and S occur, the music is sensed, the sensory representation sr(music) is generated, the preparation representation p for S is generated, S occurs, S is sensed, the sensory representation sr(S) is generated. According to Damasio (2000), pp. 177-183, two transitions are relevant (see Damasio's Figure 6.1), and have to be taken into account in a model:

- from the sensory representation of the initial no S body state and not hearing the music to hearing music and a sensory representation of the music, and no S sensory representation
- from a sensory representation of the music and no sensory representation of S to a sensory representation of S and a sensory representation of the music

These two transitions are to be detected and represented by the organism. To model this process three internal state properties are introduced: s0 for encoding the initial situation, and s1 and s2 subsequently for encoding the situations after the two relevant changes. By making these state properties persistent they play the role of indicating that in the past a certain situation has occurred. Local dynamic properties that relate these additional internal state properties to the others can be expressed as follows (see also Figure 2):

LP7 not sr(music) & not sr(S) $\bullet\!\!\rightarrow\!\!\!\rightarrow$ s0
LP8 sr(music) & not sr(S) & s0 $\bullet\!\!\rightarrow\!\!\!\rightarrow$ s1
LP9 sr(music) & sr(S) & s1 $\bullet\!\!\rightarrow\!\!\!\rightarrow$ s2

State properties s0 and s1 are persistent.

6. Simulation

A special software environment has been created to enable the simulation of executable models (Bosse et al., 2004). Based on an input consisting of dynamic properties in *leads to* format (and their timing parameters e, f, g, h, see Section 2), this software environment generates simulation traces. The algorithm used for the simulation is rather straightforward: at each time point, a bound part of the past of the trace (the maximum of all g values of all rules) determines the values of a bound range of the future trace (the maximum of f + h over all LEADSTO rules). The software was written in SWI-Prolog/XPCE, and consists of approximately 20000 lines of code. For more implementation details, see (Bosse et al., 2004).

Using this software environment, the model described in the previous sections has been used to generate a number of simulation traces. An example of such a simulation trace can be seen in Figure 3. Here, time is on the horizontal axis, the state properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false. This trace is based on all executable local properties (i.e., LP0 to LP9), except LP6. In all properties, the values (0,0,1,1) have been chosen for the timing parameters e, f, g, and h. Figure 3 shows how the presence of the music first leads to an emotion (p or S), then to a feeling (sr(S)), and finally to the birth of core consciousness (s2), involving a body loop.

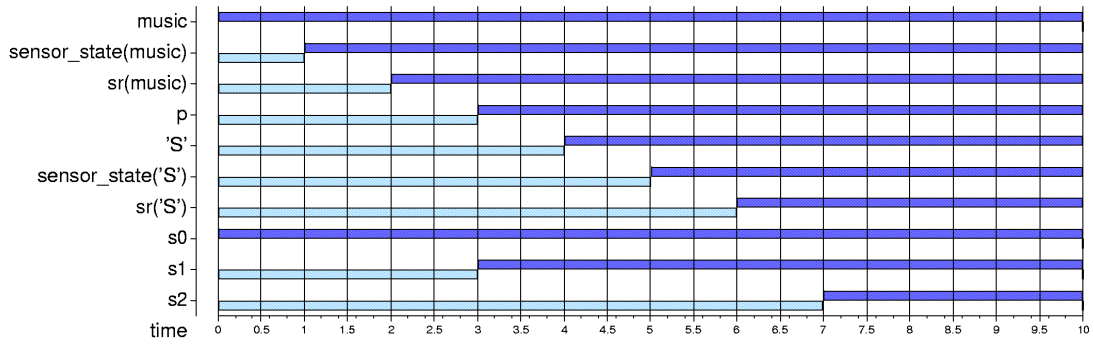


Figure 3. Simulation trace involving a body loop

A similar trace is given in Figure 4, for the case of the as-if body loop. This trace is based on all executable local properties (i.e., LP0 to LP9), except LP3, LP4, and LP5. Again, in all properties, the values (0,0,1,1) have been chosen for the timing parameters e, f, g, and h. As can be seen in Figure 4, in this case the feeling (sr(S)) immediately follows the preparatory state p, without an actual change in body state (S).

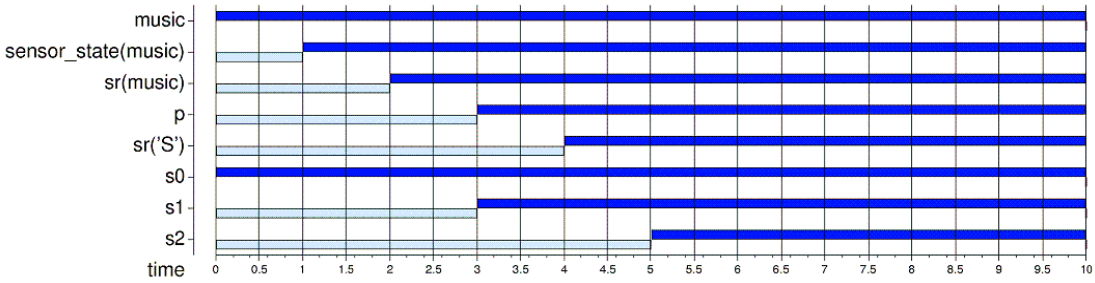


Figure 4. Simulation trace involving an as-if body loop

7. Representational Content

In Damasio's description various types of representation are used, for example, sensory representations and second-order representations. In the literature on Philosophy of Mind a number of approaches to representational content are discussed. In this section three of these approaches are briefly introduced and it is discussed in how far the types of representation used by Damasio indeed can be considered as such according to these approaches.

In (Kim, 1996), pp. 191-192 the *causal/correlational approach* to representational content is explained as follows. Suppose that, some causal chain is connecting an internal state property s and external state property 'horse nearby'. Due to this causal chain, under normal conditions internal state property s of an organism covaries regularly with the presence of a horse: this state property s occurs precisely when a horse is present nearby. Then the occurrence of s has the presence of the horse as its representational content. Especially for perceptual state properties this may work well.

In (Kim, 1996), pp. 200-202 the concept of *relational specification* of a state property is put forward as an approach to representational content. It is based on a specification of how an internal state property can be related to properties of states distant in space and time. This approach is more liberal than the causal/correlational approach, since it is not restricted to one external state, but allows reference to a whole sequence of states in history.

Finally, the *temporal-interactivist approach* (Bickhard, 1993) relates the occurrence of internal state properties to sets of past and future interaction traces. Thus, like the relational specification approach, this approach allows reference to a whole sequence of states in history (or future). However, whilst in the relational specification approach these states can have any desired type (e.g., internal, external, or interaction states), in the temporal-interactivist approach they are restricted to interaction states (i.e. observations and actions).

In the following sections it is explored whether these approaches can be used to specify the representational content of the relevant mental states that occur in our model (i.e., the states that represent emotion, feeling, and feeling a feeling). The focus is on the causal/correlational approach and the relational specification approach. The temporal-interactivist approach is not discussed. However, the formulae expressing the representational content according to the relational specification approach can be easily translated to the temporal-interactivist approach by replacing the external states that occur in the formulae by interaction states (e.g., replacing music by sensor_state(music)).

7.1. Content of Emotion

Consider the causal chain music - sensor_state(music) - sr(music) - p - S (see Figure 1). Thus, looking backward in time, the external emotional state property S can be considered to (externally) represent the emotional content of the music. On the other hand, the internal emotional state property involved is p. Given the causal chain above the (backward) representational content for both p and S is the presence of this very special music, which could be considered acceptable. However, following the same causal chain, also the state property sr(music) has the same representational content. What is different between p and sr(music)? Why are the emotional responses to the same music different between different individuals? This would not be explainable if in all cases the same representational content is assigned. It might be assumed that state properties such as sr(music) may show changes between different individuals. However, the differences are probably much larger between the ways in which for two different individuals sr(music) is connected to a composite state property p. This subjective aspect is not taken into account in the causal/correlational approach. The content of such an emotional response apparently is more personal than a reference to an objective external factor, so to define this representational content both the external music and the internal personal make up has to be taken into account.

For the relational specification approach the representational content of p can be specified in a manner similar to the causal/correlational approach by ‘p occurs if the very special music just occurred’, and conversely. However, other, more suitable possibilities are available as well, such as, ‘p occurs if the very special music just occurred, and by this organism such music was perceived as sr(music) and for this organism sr(music) leads to p’, and conversely. This relational specification involves both the external music and the internal make up of the organism, and hence provides a subjective element in the representational content, in addition to the external reference. This provides an explanation of differences in emotional content of music between individuals.

7.2. Content of Feeling

The representational content of sr(S) according to the causal/correlational approach can consider the causal chain music - sensor_state(music) - sr(music) - p - S - sensor_state(S) - sr(S). Using this chain, sr(S) can be related to both the presence of S, and further back to the presence of the very special music. This steps outside the context of having a

reference to one state, which limits the causal/correlational approach. A more suitable approach is the relational specification approach, which allows such temporal relationships to different states in the past; there is the following temporal relation between the occurrence of sr(S), the presence of the S, and the presence of music: 'sr(S) occurs if S just occurred, preceded by the presence of the music', and conversely.

7.3. Content of Feeling a Feeling

The representational content of s0 according to the causal/correlational approach can be taken as the absence of both S and music in the past, via the causal chain: no S and no music - sensor state no S and sensor state no music - no sr(music) and no sr(S) - s0. This can be expressed relationally by referring to one state in the past: 'if no S and no music occur, then later s0 will occur,' and conversely. Formally:

$\forall t1$ [state(γ, t1, EW) |== $\neg$ S $\wedge$ $\neg$ music $\Rightarrow$
$\exists t2 \geq t1$ state(γ, t2, internal) |== s0]
$\forall t2$ [state(γ, t2, internal) |== s0 $\Rightarrow$
$\exists t1 \leq t2$ state(γ, t1, EW) |== $\neg$ S $\wedge$ $\neg$ music]

For s1 and s2 the causal/correlational approach does not work very well because these state properties essentially encode (short) histories of states. For example, the representational content of s1 according to causal/ correlational approach can be tried as follows: presence of the music and no S in the past under the condition that at some point in time before that point in time no music occurred. However, this cannot be expressed adequately according to the causal/ correlational approach since it is not one state in the past to which reference is made, but a history given by some temporal sequence. The problem is that no adequate solution is possible, since the internal state properties should in fact be related to sequences of different inputs over time in the past. This is something the causal/correlational approach cannot handle, as reference has to be made to another state at one time point, and it is not possible to refer to histories, i.e., sequences of states over time, in the past. A better option is provided by representational content of s1 as relational specification: 'if no S and no music occur, and later music occurs and still no S occurs, then still later s1 will occur,' and conversely. Formally:

$\forall t1, t2$ [$t1 \leq t2$ & state(γ, t1, EW) |== $\neg$ S $\wedge$ $\neg$ music &
state(γ, t2, EW) |== $\neg$ S $\wedge$ music $\Rightarrow$
$\exists t3 \geq t2$ state(γ, t3, internal) |== s1]
$\forall t3$ [state(γ, t3, internal) |== s1 $\Rightarrow$ $\exists t1, t2$ $t1 \leq t2 \leq t3$ &
state(γ, t1, EW) |== $\neg$ S $\wedge$ $\neg$ music &
state(γ, t2, EW) |== $\neg$ S $\wedge$ music]

Similarly, the representational content of s2 as relational specification can be specified as follows: 'if no S and no music occur, and later music occurs and still no S occurs, and later music occurs and S occurs, then still later s2 will occur,' and conversely. Formally:

$\forall t1, t2, t3$ [$t1 \leq t2 \leq t3$ & state(γ, t1, EW) |== $\neg$ S $\wedge$ $\neg$ music &
state(γ, t2, EW) |== $\neg$ S $\wedge$ music &
state(γ, t3, EW) |== S $\wedge$ music $\Rightarrow$
$\exists t4 \geq t3$ state(γ, t4, internal) |== s2]
$\forall t4$ [state(γ, t4, internal) |== s2 $\Rightarrow$
$\exists t1, t2, t3$ $t1 \leq t2 \leq t3 \leq t4$ &
state(γ, t1, EW) |== $\neg$ S $\wedge$ $\neg$ music &
state(γ, t2, EW) |== $\neg$ S $\wedge$ music &
state(γ, t3, EW) |== S $\wedge$ music]

This comes close to the transitions mentioned in Section 5: the proto-self exists at the inaugural instant - an object comes into sensory representation - the proto-self has become modified by the object.

The above relational specification is a first-order representation in that it refers to external states of world and body, whereas Damasio's second-order representation refers to internal states (other, first-order, representations) of the proto-self. The relational specification given above only works for body loops, not for 'as if body loops'. A relational specification that comes more close to Damasio's formulation, and also works for 'as if body loops' is the following (RSP):

$\forall t1, t2, t3$ [$t1 \leq t2 \leq t3$ &
  state(γ, t1, internal) |== $\neg$ sr(S) $\wedge$ $\neg$ sr(music) &
  state(γ, t2, internal) |== $\neg$ sr(S) $\wedge$ sr(music) &
  state(γ, t3, internal) |== sr(S) $\wedge$ sr(music) $\Rightarrow$
    $\exists t4 \geq t3$ state(γ, t4, internal) |== s2]
$\forall t4$ [state(γ, t4, internal) |== s2 $\Rightarrow$
  $\exists t1, t2, t3$ $t1 \leq t2 \leq t3 \leq t4$ &
  state(γ, t1, internal) |== $\neg$ sr(S) $\wedge$ $\neg$ sr(music) &
  state(γ, t2, internal) |== $\neg$ sr(S) $\wedge$ sr(music) &
  state(γ, t3, internal) |== sr(S) $\wedge$ sr(music)]

This is a relational specification in terms of other representations (sr(music), sr(S)), and therefore a second-order representation. It has no direct reference to external states anymore. However, indirectly, via the first-order representations sr(music) and sr(S) it has references to external states.

8. Verification

In Sections 3-6, local, executable dynamic properties were addressed, and simulation based on these properties was discussed. In Section 7, dynamic properties to describe representational content of internal states are introduced. These dynamic properties are of a *global* nature. Another example of a more global property is the following:

OP1 music $\bullet\!\!\rightarrow\!\!\!\!\rightarrow$ s2

Informally, this property states that the presence of music eventually leads to the birth of core consciousness (s2). This can be considered as a global property because it describes dynamic of the overall process, whereas the properties presented in Sections 3-6 described basic steps of the process. For both types of global properties (i.e., dynamic property OP1 and the properties specifying representational content), an important issue is *verification*. In other words, are these global properties satisfied by the simulation model described in Sections 3-6? Therefore, the global properties have been formalised, and verification has been applied in two ways: by *automated checks* and by establishing *logical relationships*.

8.1. Automated Checks

In addition to the simulation software described in Section 6, a software environment has been developed that enables to check dynamic properties specified in TTL against simulation traces. This software environment takes a dynamic property and one or more (empirical or simulated) traces as input, and checks whether the dynamic property holds for the traces. Using this environment, the global properties mentioned above have been

automatically checked against traces like depicted in Figure 3 and 4. The duration of these checks varied between 0.5 and 1.5 seconds, depending on the complexity of the formula. All these checks turned out to be successful, which validates (for the given traces at least) our choice for the representational content of the internal state properties. However, note that these checks are only an empirical validation, they are no exhaustive proof as, e.g., model checking is.

8.2. Logical Relationships

A second way of verification is to establish logical relationships between global properties and local properties. This has been performed in a number of cases. For example, to relate OP1 to local properties, intermediate properties were identified in the form of the following milestone properties that split up the process in three phases:

MP1(MtoE)	music $\bullet\!\!\rightarrow\!\!\!\!\rightarrow$ sr(music) & sr(music) $\bullet\!\!\rightarrow\!\!\!\!\rightarrow$ S
MP2(EtoF)	S $\bullet\!\!\rightarrow\!\!\!\!\rightarrow$ sr(S)
MP3(FtoFF)	**RSP** (see Section 7)

For the milestone properties the following relationships hold (for simplicity neglecting 'as if body loops'):

MP1(MtoE) & MP2(EtoF) & MP3(FtoFF)	$\Rightarrow$	OP1
LP0 & LP1 & LP2 & LP3	$\Rightarrow$	MP1(MtoE)
LP4 & LP5	$\Rightarrow$	MP2(EtoF)
LP7 & LP8 & LP9	$\Rightarrow$	MP3(FtoFF)

Figure 5 provides the same relationships in the form of a logical AND-tree.

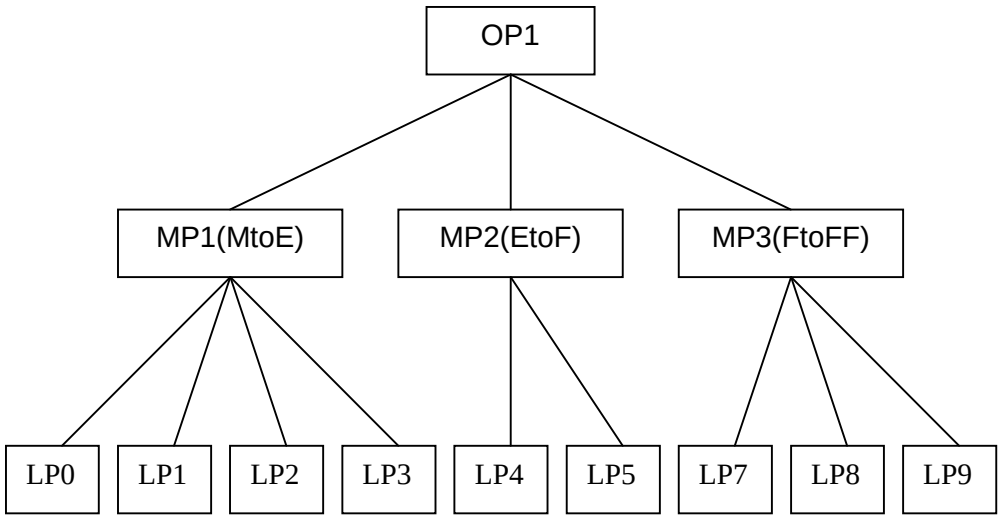


Figure 5. Logical relationships between the dynamic properties

Such logical relationships between properties can be very useful in the analysis of traces. For example, if a given trace that is unsuccessful does not satisfy milestone property MP2, then by a refutation process it can be concluded that the cause can be found in either LP4 or LP5. In other words, either the sensor mechanism fails (LP4), or the sensory representation mechanism fails (LP5).

9. Discussion

The chosen modelling approach describes temporal dependencies in processes at a neurological, not symbolic level. To avoid complexity the model is specified at an abstract level. From the available approaches to representational content from Philosophy of Mind, the causal/correlational approach is not applicable, but Kim's relational specification approach, that allows more complex temporal dependencies, is applicable. Using this approach, claims on representational content made by Damasio have been formalised and supported by means of verification.

Furthermore, an interesting observation that has been made on the basis of the formalisation was that the model predicted the possibility of 'false core consciousness': core consciousness that is attributed to the 'wrong' stimulus. To explain this phenomenon, suppose that two stimuli occur, say x1 and x2, where x2 is subliminal and unnoticed. Then, it could be the case that x2 provokes emotional responses, whilst the conscious feeling that arises is attributed to x1 instead of x2. In terms of our model, this can be simulated by first introducing a subliminal stimulus that yields emotion S (e.g., a cold breeze) followed by the stimulus music. In that case, the conscious feeling would incorrectly be attributed to the music. In personal communication with Antonio Damasio, the existence of this predicted false core consciousness was confirmed.

For the philosophical perspective the paper contributes a case study for representational content which is more down-to-earth than the science fiction style thought experiments, such as the planet Twin Earth, that are common in the literature on Philosophy of Mind, e.g., (Kim, 1996). In addition, the type of representation is more sophisticated than the usual ones essentially addressing sensory representations induced by observing (a snapshot of) a horse or a tomato. Interesting further work in this area is to analyse various arguments given in this literature by applying them to this example.

The analysis approach that is applied in this paper to model Damasio's theory of consciousness, has previously been applied to complex and dynamic cognitive processes other than consciousness, such as the interaction between agent and environment (Bosse, Jonker, and Treur, 2004). In a number of these cases, in addition to simulated traces, also empirical (human) traces have been formally analysed. Using this approach, it is possible to verify global dynamic properties (e.g., specifying the representational content of internal states) in real-world situations.

For recent work in the area of emotion and consciousness, the interested reader is referred to (Prinz and Chalmers, 2004), Chapter 3, which gives an account for emotions as embodied representations of "core relational themes" such as danger and obstruction.

Acknowledgements

The authors are grateful to Antonio Damasio for his valuable comments upon their questions, and to an anonymous referee for some suggestions for improvements of an earlier version of this paper.

References

Bickhard, M.H., Representational Content in Humans and Machines. *Journal of Experimental and Theoretical Artificial Intelligence*, 5, 1993, pp. 285-333.

Bosse, T., Jonker, C. M., van der Meij, L., and Treur, J., LEADSTO: a Language and Environment for Analysis of Dynamics by SimulaTiOn. Vrije Universiteit Amsterdam, Department of Artificial Intelligence. Technical Report, 2004.

Bosse, T., Jonker, C.M., and Treur, J., *Representational Content and the Reciprocal Interplay of Agent and Environment* In: Leite, J., Omincini, A., Torroni, P., and Yolum, P. (eds.), Proceedings of the Second International Workshop on Declarative Agent Languages and Technologies, DALT'04, Springer Verlag, 2004, pp. 61-76.

Clark, A., *Being There: Putting Brain, Body and World Together Again*. MIT Press, 1997.

Damasio, A., *The Feeling of What Happens: Body, Emotion and the Making of Consciousness*. MIT Press, 2000.

Damasio, A., Grabowski, T.J., Bechara, A., Damasio, H., Ponto, L.L.B., Parvizi, J., and Hichwa, R.D., Subcortical and cortical brain activity during the feeling of self-generated emotions. *Nature Neuroscience*, vol. 3, 2000, pp. 1049-1056.

Jonker, C.M. and Treur, J., Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. *International Journal of Cooperative Information Systems*, vol. 11, 2002, pp. 51-92.

Jonker, C.M. and Treur, J., A Temporal-Interactivist Perspective on the Dynamics of Mental States. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 137-155.

Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 191-210.

Kim, J., *Philosophy of Mind*. Westview Press, 1996.

Parvizi, J. and Damasio, A., Consciousness and the brain stem. *Cognition*, vol. 79, 2001, pp. 135-159.

Prinz, J.J. Chalmers, D.J., Gut Reactions: A Perceptual Theory of Emotion (Philosophy of Mind (Hardcover)), Oxford University Press, 2004.

PART VI REPRESENTATIONAL CONTENT

CHAPTER 19

Modelling Shared Extended Mind and Collective Representational Content

Part of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2004). Modelling Shared Extended Mind and Collective Representational Content. In: Bramer, M., Coenen, F., and Allen, T. (eds.), *Research and Development in Intelligent Systems XXI, Proceedings of AI-2004, the 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Springer Verlag, pp 19-32.

Part of this chapter appeared as Bosse, T., Jonker, C.M., and Treur, J. (2004). Modelling Shared Extended Mind and Collective Representational Content. In: Castelfranchi, C. (ed.), *Proceedings of the Fourth Conference on Collective Intentionality*.

Modelling Shared Extended Mind and Collective Representational Content

Tibor Bosse[1], Catholijn M. Jonker[1,2], Martijn C. Schut[1], and Jan Treur[1,3]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: {tbosse, schut, treur}@cs.vu.nl, C.Jonker@nici.ru.nl
URLs: http://www.cs.vu.nl/~{tbosse, jonker, schut, treur}
[2] Radboud Universiteit Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
[3] Universiteit Utrecht, Department of Philosophy,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Abstract. Some types of animals exploit the external environment to support their cognitive processes, in the sense of patterns created in the environment that function as external mental states and serve as an extension to their mind. In the case of social animals the creation and exploitation of such patterns can be shared, thus obtaining a form of shared mind or collective intelligence. This paper explores this shared extended mind principle for social animals in more detail. The focus is on the notion of representational content in such cases. Proposals are put forward and formalised to define collective representational content for such shared external mental states. A case study in social ant behaviour in which shared extended mind plays an important role is used as illustration. For this case simulations are described, representation relations are specified and are verified against the simulated traces.

1. Introduction

Behaviour is often not only supported by internal mental structures and cognitive processes, but also by processes based on patterns created in the external environment that serve as external mental structures; cf. (Clark, 1997, 2001; Clark and Chalmers, 1998, Dennett, 1996). Examples of this pattern of behaviour are the use of 'to do lists' and 'lists of desiderata'. Having written these down externally (e.g., on paper, in your diary, in your organizer or computer) makes it unnecessary to have an internal memory about all the items. Thus internal mental processing can be kept less complex. Other examples of the use of extended mind are doing mathematics or arithmetic, where external (symbolic, graphical, material) representations are used; e.g., (Bosse et al., 2003). In (Menary, 2004) a collection of papers can be found based on presentations at the conference 'The Extended Mind: The Very Idea' that took place in 2001. Clark (2001) points at the roles played by both internal and external representations in describing cognitive processes:

> *'Internal representations will, almost certainly, feature in this story. But so will external representations, …' (Clark, 2001, p. 134).*

From another, developmental angle, also Griffiths and Stotz (2000) endorse the importance of using both internal and external representations; they speak of

> *'a larger representational environment which extends beyond the skin'*

and claim that

> *'culture makes humans as much as the reverse' (Griffiths and Stotz, 2000, p. 45).*

Allowing mental states, which are in the external world and thus accessible for any agent around, opens the possibility that other agents also start to use them. Indeed, not only in the individual, single agent case, but also in the social, multi-agent case the extended mind principle can be observed, e.g., one individual creating a pattern in the environment, and one or more other individuals taking this pattern into account in their behaviour. For the human case, examples can be found everywhere, varying from roads, and traffic signs to books or other media, and to many other kinds of cultural achievements. Also in (Scheele, 2002) it is claimed that part of the total team knowledge in distributed tasks (such as air traffic control) comprises external memory in the form of artefacts. In this multi-agent case the extended mind principle serves as a way to build a form of social or collective intelligence, that goes beyond (and may even not require) social intelligence based on direct one-to-one communication.

Especially in the case of social animals external mental states created by one individual can be exploited by another individual, or, more general, the creation and maintenance, as well as the exploitation of external mental states can be activities in which a number of individuals participate. For example, presenting slides on a paper with multiple authors to an audience. In such cases the external mental states cross, and in a sense break up, the borders between the individuals and become *shared extended mental states*. An interesting and currently often studied example of collective intelligence is the intelligence shown by an ant colony (Bonabeau et al., 1999). Indeed, in this case the external world is exploited as an extended mind by using pheromones. While they walk, ants drop pheromones on the ground. The same or other ants sense these pheromones and follow the route in the direction of the strongest sensing. Pheromones are not persistent for long times; therefore such routes can vary over time.

In (Bosse et al., 2004) the shared extended mind principle is worked out in more detail. The paper focusses on formal analysis and formalisation of the dynamic properties of the processes involved, both at the local level (the basic mechanisms) and the global level (the emerging properties of the whole), and their relationships. A case study in social ant behaviour in which shared extended mind plays an important role is used as illustration.

In the current paper, as an extension to (Bosse et al., 2004), the notion of *representational content* is analysed for mental processes based on the shared extended mind principle. The analysis of notions of representational content of *internal* mental state properties is well-known in the literature on Cognitive Science and Philosophy of Mind. In this literature a relevant internal mental state property m is taken and a representation relation is identified that indicates in which way m relates to properties in the external world or the agent's interaction with the external world; cf. (Bickhard, 1993; Jacob, 1997, Kim, 1996, pp. 184-210). For the case of extended mind an extension of the analysis of notions of representational content to *external* state properties is needed. Moreover, for the case of external mental state properties that are *shared*, a notion of *collective* representational content is needed (in contrast to a notion of representational content for a single agent).

Thus, by addressing the ants example and its modelling from an extended mind perspective, a number of challenging new issues on cognitive modelling and representational content are encountered:

- How to define representational content for an *external* mental state property
- How to handle *decay* of a mental state property
- How can *joint* creation of a *shared* mental state property be modelled

- What is an appropriate notion of *collective* representational content of a shared external mental state property
- How can representational content be defined in a case where a behavioural choice depends on *a number of mental state properties*

In this paper these questions are addressed. To this end the shared extended mind principle is analysed in more detail, and a formalisation is provided of its dynamics. It is discussed in particular how a notion of collective representational content for a shared external mental state property can be formulated. In the literature notions of representational content are usually restricted to internal mental states of one individual. The notion of collective representational content developed here extends this in two manners: (1) for external instead of internal mental states, and (2) for groups of individuals instead of single individuals. It is reported how in a case study of social behaviour based on shared extended mind (a simple ant colony) the proposals put forward have been evaluated. The analysis of this case study comprises multi-agent simulation based on identified local dynamic properties, identification of dynamic properties that describe collective representational content of shared extended mind states, and verification of these dynamic properties.

2. State Properties and Dynamic Properties

Dynamics will be described in the next section as evolution of states over time. The notion of state as used here is characterised on the basis of an ontology defining a set of physical and/or mental (state) properties that do or do not hold at a certain point in time. For example, the internal state property 'the agent A has pain', or the external world state property 'the environmental temperature is 7° C', may be expressed in terms of different ontologies. To formalise state property descriptions, an ontology is specified as a finite set of sorts, constants within these sorts, and relations and functions over these sorts. The example properties mentioned above then can be defined by nullary predicates (or proposition symbols) such as pain, or by using n-ary predicates (with $n \geq 1$) like has_temperature(environment, 7). For a given ontology Ont, the propositional language signature consisting of all *state ground atoms* (or *atomic state properties*) based on Ont is denoted by APROP(Ont). The *state properties* based on a certain ontology Ont are formalised by the propositions that can be made (using conjunction, negation, disjunction, implication) from the ground atoms. A *state* S is an indication of which atomic state properties are true and which are false, i.e., a mapping S: APROP(Ont) $\rightarrow$ {true, false}.

To describe the internal and external dynamics of the agent, explicit reference is made to time. Dynamic properties can be formulated that relate a state at one point in time to a state at another point in time. A simple example is the following dynamic property specification for belief creation based on observation:

'at any point in time t1, if the agent observes at t1 that it is raining,
then there exists a point in time t2 after t1 such that at t2 the agent believes that it is raining'.

To express such dynamic properties, and other, more sophisticated ones, the temporal trace language TTL is used; cf. (Jonker et al., 2003). To express dynamic properties in a precise manner a language is used in which explicit references can be made to time points and traces. Here *trace or trajectory* over an ontology Ont is a time-indexed sequence of states over Ont. The sorted predicate logic temporal trace language TTL is built on atoms referring to, e.g., traces, time and state properties. For example, 'in the output state of A in trace γ at time t property p holds' is formalised by state(γ, t, output(A)) |= p. Here |= is a

predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus. Dynamic properties are expressed by temporal statements built using the usual logical connectives and quantification (for example, over traces, time and state properties). For example the following dynamic property is expressed:

'in any trace γ, if at any point in time t1 the agent A observes that it is raining,
then there exists a point in time t2 after t1 such that
at t2 in the trace the agent A believes that it is raining'.

In formalised form:

$\forall t1$ [state(γ, t1, input(A)) |= agent_observes_itsraining $\Rightarrow$
$\exists t2 \geq t1$ state(γ, t2, internal(A)) |= belief_itsraining]

Language abstractions by introducing new (definable) predicates for complex expressions are possible and supported.

A simpler temporal language has been used to specify simulation models. This language (the *leads to* language) offers the possibility to model direct temporal dependencies between two state properties in successive states. This executable format is defined as follows. Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the *leads to* language the notation $\alpha \rightarrow\!\!\!\rightarrow_{e, f, g, h} \beta$ means:

If state property α holds for a certain time interval with duration g, then after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see (Jonker et al., 2003). A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

3. Representation for Shared Extended Mind

Originally, the different types of approaches to representational content that have been put forward in the literature on Cognitive Science and Philosophy of Mind, (Bickhard, 1993; Jonker and Treur, 2003; Kim, 1996, pp. 191-193, 200-202) are applicable to internal (mental) states. They have in common that the occurrence of the internal (mental) state property m at a specific point in time is related (by a representation relation) to the occurrence of other state properties, at the same or at different time points. For the *temporal-interactivist* approach (Bickhard, 1993; Jonker and Treur, 2003) a representation relation relates the occurrence of an internal state property to sets of past and future interaction traces. The *relational specification* approach to representational content is based on a specification of how a representation relation relates the occurrence of an internal state property to properties of states distant in space and time; cf. (Kim, 1996, pp. 200-202). As mentioned in the Introduction, one of the goals of this paper is to apply these approaches to *shared extended* mental states instead of *internal* mental states.

Suppose p is an external state property used by a collection of agents in their shared extended mind, for example, as an external belief. At a certain point in time this mental state property is created by performing an action a (or maybe a collection of actions) by one or more agents to bring about p in the external world. Given the thus created occurrence of p, at a later point in time any agent can observe p and take this mental state property into account in determining its behaviour. For a representation relation, which

indicates representational content for such a mental state property p two possibilities are considered:

- a representation relation relating the occurrence of p to one or more events in the past (*backward*)
- a representation relation relating the occurrence of p to behaviour in the future (*forward*)

Moreover, for each category, the representation relation can be described by referring to:

- interaction state properties (e.g., observing, initiating actions) for the agent (e.g., using the *temporal-interactivist* approach)
- external world state properties, independent of the agent (e.g., using the *relational specification* approach)

A final distinction between representation relations depends on the nature of mental state property p:

- *the qualitative case*. Here p may be the result of the action of one agent (e.g., p is the presence of pheromone)
- *the quantitative case*. Here p may be the result of actions of multiple agents. Here p has a certain degree or level (e.g., a certain accumulated level of pheromone); in decisions levels for a number of such state properties p may be taken into account

So, eight types of approaches (2x2x2) to representational content are distinguished. These approaches will be presented in the next two sections. Table 1 gives an overview about which approach is given in which section.

	backward temporal-interactivist approach	backward relational specification approach	forward temporal-interactivist approach	forward relational specification approach
qualitative case	4.1	4.2	4.3	4.4
quantitative case	5.1	5.2	5.3	5.4

Table 1. Different types of Representation Relations

4. The Qualitative Case

For the ants case study, the world in which the ants live is described by a labeled graph as depicted in Figure 1. Locations are indicated by A, B,…, and edges by e1, e2,… To represent such a graph the predicate connected_to_via(l0,l1,e) is used. The ants move from location to location via edges; while passing an edge, pheromones are dropped. The objective of the ants is to find food and bring this back to the nest. In this example there is only one nest (at location A) and one food source (at location F).

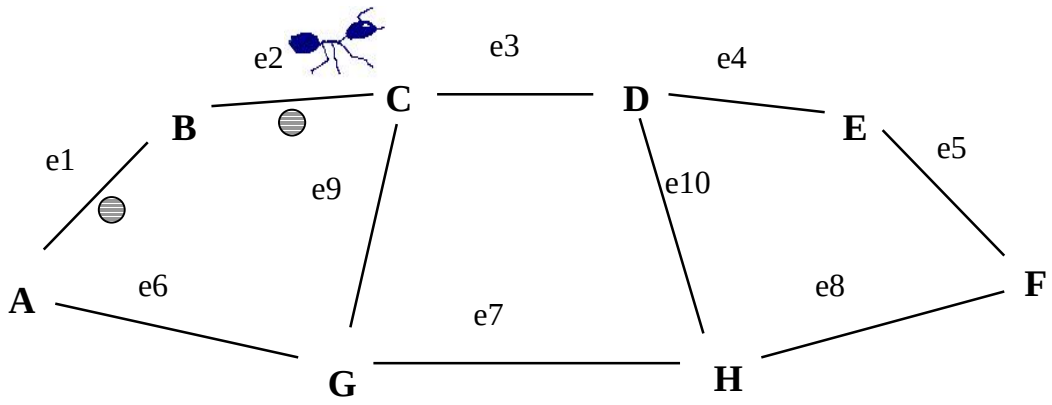


Figure 1. An ants world

In this section representational content is addressed for the qualitative case. This means that an external state property p (e.g., the presence of pheromone) is the result of the action of one agent (e.g., dropping the pheromone).

4.1. Backward Temporal-Interactivist Approach

Looking backward, for the qualitative case the preceding state is the action a by an arbitrary agent, to bring about p. This action a is an interaction state property of the agent. Thus, for the temporal-interactivist approach a representation relation can be specified by temporal relationships between p (the presence of the pheromone at a certain edge), and a (the action of dropping this pheromone). In an informal notation, this representation relation look as follows:

If at some time point in the past an agent dropped pheromone at edge e,
then after that time point the pheromone was present at edge e.

If the pheromone is present at edge e,
then at some time point in the past an agent dropped it at e.

Note here that the sharing of the external mental state property is expressed by using explicit agent names in the language and quantification over (multiple) agents. In the usual single agent case of a representation relation, no explicit reference to the agent itself is made. A formalisation is as follows:

$\forall t1\ \forall l\ \forall e\ \forall a$ [state(γ, t1) |=
to_be_performed(a, drop_pheromones_at_edge_from(e, l))
$\Rightarrow \exists t2{>}t1$ state(γ, t2) |= pheromone_at(e)]

$\forall t2\ \forall e$ [state(γ, t2) |= pheromone_at(e)
$\Rightarrow \exists a, l, t1{<}t2$ state(γ, t1) |=
to_be_performed(a, drop_pheromones_at_edge_from(e, l))]

4.2. Backward Relational Specification Approach

As mentioned above, the action of dropping pheromone is not an external state property but an interaction state property of this agent. However, this action was performed due to certain circumstances in the world that made the agent do the action. So, the chain of processes can be followed further back to the agent's internal state properties. Still further back it can be followed to the agent's observations that in the past formed the basis of these internal state properties. As these observations concern observations of certain state

properties of the external world, we finally arrive at other external world state properties. These external world state properties will be used for the representation relation conform the relational specification approach. It may be clear that if complex internal processes come between, such a representation relation can become complicated. However, if the complexity of the agent's internal processes is kept relatively simple (as is one of the claims accompanying the extended mind principle), this amounts in a feasible approach.

For the relational specification approach a representation relation can be specified by temporal relationships between the presence of the pheromone (at a certain edge), and other state properties in the past or future. Although the relational specification approach as such does not explicitly exclude the use of state properties related to input and output of the agent, in our approach below the state properties will be limited to external world state properties. As the mental state property itself also is an external world state property, this implies that temporal relationships are provided only between external world state properties. The pheromone being present at edge e is temporally related to the existence of a state at some time point in the past, namely an agent's presence at e:

If at some time point in the past an agent was present at e,
then after that time point the pheromone was present at edge e.

If the pheromone is present at edge e,
then at some time point in the past an agent was present at e,

A formalisation is as follows:

$\forall t1\ \forall l\ \forall l1\ \forall e\ \forall a$ [state(γ, t1) |= is_at_edge_from_to(a, e, l, l1)
$\Rightarrow \exists t2{>}t1$ state(γ, t2) |= pheromone_at(e)]

$\forall t2\ \forall e$ [state(γ, t2) |= pheromone_at(e)
$\Rightarrow \exists a, l, l1, t1{<}t2$ state(γ, t1) |= is_at_edge_from_to(a, e, l, l1)]

4.3. Forward Temporal-Interactivist Approach

Looking forward, in general the first step is to relate the extended mind state property p to the observation of it by an agent (under certain circumstances c). However, again the chain of processes can be followed further through this agent's internal processes to the agent's actions (for the temporal-interactivist approach) and their effects on the external world (for the relational specification approach).

For the example, an agent's action based on its observation of the pheromone is that it heads for the direction of the pheromone. So, according to the temporal-interactivist approach, the representation relation relates the occurrence of the pheromone (at edge e) to the conditional (with condition that it observes the location) fact that the agent heads for the direction of e. The pheromone being present at edge e is temporally related to a conditional statement about the future, namely if an agent later observes the location, coming from any direction e', then he will head for direction e:

If the pheromone is present at edge e1,
then if at some time point in the future, an agent observes a location l, connected to e1, coming from any direction e2≠e1,
then the next direction he will choose is e1.

If a time point t1 exist such that
at t1 an agent observes a location l (connected to e1), coming from any direction e2 ≠e1,
and if at any time point t2 ≥ t1 an agent observes this location l coming from any direction e3≠e1,
then the next direction he will choose is e1,
then at t1 the pheromone is present at direction e1.

A formalisation is as follows:

∀t1 ∀l ∀l1 ∀e1 [state(γ, t1) |= pheromone_at(e1) ⇒
∀t2>t1 ∀e2, a
[e2 ≠ e1 & state(γ, t2) |= connected_to_via(l, l1, e1) &
state(γ, t2) |= observes(a, is_at_location_from(l, e2))
⇒ ∃t3>t2 state(γ, t3) |=
to_be_performed(a, go_to_edge_from_to(e1, l, l1)) &
[∀t4 t2<t4<t3 ⇒ observes(a, is_at_location_from(l, e2))]]]

∀t1 ∀l ∀e1 [∃a, e2 e2 ≠ e1 &
state(γ, t1) |= observes(a, is_at_location_from(l, e2)) &
[∀t2≥t1 ∀a, e3 [e3 ≠ e1 &
state(γ, t2) |= observes(a, is_at_location_from(l, e3)) ⇒
∃t3>t2 ∃l1 state(γ, t3) |=
to_be_performed(a, go_to_edge_from_to(e1, l, l1)) &
[∀t4 t2<t4<t3 ⇒ observes(a, is_at_location_from(l, e3))]]]
⇒ state(γ, t1) |= pheromone_at(e1)]

4.4. Forward Relational Specification Approach

The effect of an agent's action based on its observation of the pheromone is that it is at the direction of the pheromone. So, according to the relational specification approach the representation relation relates the occurrence of the pheromone (at edge e) to the conditional (with condition that it is at the location) fact that the agent arrives at edge e. The pheromone being present at edge e is temporally related to a conditional statement about the future, namely if an agent arrives at the location, coming from any direction e', then later he will be at edge e:

If the pheromone is present at edge e1,
then if at some time point in the future, an agent arrives at a location l, connected to e1, coming from any direction e2≠e1,
then the next edge he will be at is e1.

If a time point t1 exist such that
at t1 an agent arrives at a location l (connected to e1), coming from any direction e2 ≠e1,
and if at any time point t2 ≥ t1 an agent arrives at this location l coming from any direction e3≠e1,
then the next edge he will be at is e1,
then at t1 the pheromone is present at direction e1.

A formalisation is as follows:

∀t1 ∀l ∀l1 ∀e1 [state(γ, t1) |= pheromone_at(e1) ⇒
∀t2>t1 ∀e2, a
[e2 ≠ e1 & state(γ, t2) |= connected_to_via(l, l1, e1) &
state(γ, t2) |= is_at_location_from(a, l, e2) ⇒
∃t3>t2 state(γ, t3) |= is_at_edge_from_to(a, e1, l, l1) &
[∀t4 t2<t4<t3⇒ is_at_location_from(a, l, e2)]]]

∀t1 ∀l ∀e1 [∃a, e2 e2 ≠ e1 &
state(γ, t1) |= is_at_location_from(a, l, e2) &
[∀t2≥t1 ∀a, e3 [e3 ≠ e1 &
state(γ, t2) |= is_at_location_from(a, l, e3) ⇒
∃t3>t2 ∃l1 state(γ, t3) |= is_at_edge_from_to(a, e1, l, l1) &
[∀t4 t2<t4<t3 ⇒ is_at_location_from(a, l, e3)]]]
⇒ state(γ, t1) |= pheromone_at(e1)]

5. The Quantitative Case

The quantitative, accumulating case allows us to consider certain levels of a mental state property p; in this case a mental state property is involved that is parameterised by a number: it has the form p(r), where r is a number, denoting that p has level r. This differs from the above in that now the following aspects have to be modeled: (1) joint creation of p: multiple agents together bring about a certain level of p, each contributing a part of the level, (2) by decay, levels may decrease over time, (3) behaviour may be based on a number of state properties with different levels, taking into account their relative values, e.g., by determining the highest level of them. For the ants example, for each choice point multiple directions are possible, each with a different pheromone level; the choice is made for the direction with the highest pheromone level (ignoring the direction the ant just came from).

5.1. Backward Temporal-Interactivist Approach

To address the backward quantitative case (i.e., the case of joint creation of a mental state property), the representation relation is analogous to the one described in Section 4, but now involves not the presence of one agent at one past time point, but a summation over multiple agents at different time points. Moreover a decay rate r with $0 < r < 1$ is used to indicate that after each time unit only a fraction r is left.

For the ants example in mathematical terms the following property is expressed (according to the temporal-interactivist approach):

There is an amount v of pheromone at edge e, if and only if there is a history such that at time point 0 there was ph(0, e) pheromone at e, and for each time point k from 0 to t a number dr(k, e) of ants dropped pheromone, and $v = ph(0, e) * r^t + \Sigma_{k=0}^{t} dr(t-k, e) * r^k$

A formalisation of this property in the logical language TTL is as follows:

$$\forall t\ \forall e\ \forall v\ \ state(\gamma, t) \models pheromones\_at(e, v) \Leftrightarrow$$
$$\Sigma_{k=0}^{t} \Sigma_{a=ant1}^{ants} case(state(\gamma, k) \models to\_be\_performed(a, drop\_pheromones\_at\_edge(e)), 1, 0) * r^{t-k} = v$$

Here for any formula f, the expression case(f, v1, v2) indicates the value v1 if f is true, and v2 otherwise.

5.2. Backward Relational Specification Approach

Using the relational specification approach, the only difference is that the ants' actions of dropping pheromones at the edge are replaced by their presence at the edge:

There is an amount v of pheromone at edge e, if and only if there is a history such that at time point 0 there was ph(0, e) pheromone at e, and for each time point k from 0 to t a number dr(k, e) of ants was present at e, and $v = ph(0, e) * r^t + \Sigma_{k=0}^{t} dr(t-k, e) * r^k$

A formalisation of this property in the logical language TTL is as follows:

$$\forall t\ \forall e\ \forall v\ \ state(\gamma, t) \models pheromones\_at(e, v) \Leftrightarrow$$
$$\Sigma_{k=0}^{t} \Sigma_{a=ant1}^{ants} case(state(\gamma, k) \models is\_at\_edge(a, e), 1, 0) * r^{t-k} = v$$

5.3. Forward Temporal-Interactivist Approach

The forward quantitative case involves a behavioural choice that depends on the relative levels of multiple mental state properties. This makes that at each choice point the

representational content of the level of one mental state property is not independent of the level of the other mental state properties involved at the same choice point. Therefore it is only possible to provide representational content for the combined mental state property involving all mental state properties involved in the behavioural choice.

For the ants example the following property is specified according to the temporal-interactivist approach:

If at time t1 the amount of pheromone at edge e1 (connected to location l) is maximal with respect to the amount of pheromone at all other edges connected to that location l, except the edge that brought the ant to the location,
then, if an ant observes that location l at time t1,
then the next direction the ant will choose at some
time t2 > t1 is e1.

If at time t1 an ant observes location 1 and
for every ant observing that location 1 at time t1,
the next direction it will choose at some time t2 > t1 is e1,
then the amount of pheromone at edge e1 is maximal with respect to the amount of pheromone at all other edges connected to that location l, except the edge that brought the ant to the location.

A formalisation of this property in TTL is as follows:

$\forall$t1,l,l1,e1,e2,i1
[e1$\neq$e2 &
state(γ, t1) |= connected_to_via(l, l1, e1) &
state(γ, t1) |= pheromones_at(e1, i1) &
[$\forall$l2$\neq$l1, e3$\neq$e2 [state(γ, t1) |= connected_to_via(l, l2, e3) $\Rightarrow$
$\exists$i2 [0$\leq$i2<i1 & state(γ, t1) |= pheromones_at(e3, i2)]]
$\Rightarrow$ $\forall$a [state(γ, t1) |= observes(a, is_at_location_from(l, e2)) $\Rightarrow$
$\exists$t2>t1 state(γ, t2) |=
to_be_performed(a, go_to_edge_from_to(e1, l, l1)) &
[$\forall$t3 t1<t3<t2 $\Rightarrow$ observes(a, is_at_location_from(l, e2))]]]]

$\forall$t1, l,l1,e1,e2
[e1$\neq$e2 &
state(γ, t1) |= connected_to_via(l, l1, e1) &
$\exists$a state(γ, t1) |= observes(a, is_at_location_from(l, e2)) &
$\forall$a [state(γ, t1) |= observes(a, is_at_location_from(l, e2)) $\Rightarrow$
$\exists$t2>t1 state(γ, t2) |=
to_be_performed(a, go_to_edge_from_to(e1, l, l1)) &
[$\forall$t3 t1<t3<t2 $\Rightarrow$ observes(a, is_at_location_from(l, e2))]]]
$\Rightarrow$ $\exists$i1 [state(γ, t1) |= pheromones_at(e1, i1) &
[$\forall$l2$\neq$l1, e3$\neq$e2 [state(γ, t1) |= connected_to_via(l, l2, e3)
$\Rightarrow$ $\exists$i2 [0$\leq$i2$\leq$i1 & state(γ, t1) |= pheromones_at(e3, i2)]]]]

5.4. Forward Relational Specification Approach

Likewise, according to the relational specification approach the following property is specified:

If at time t1 the amount of pheromone at edge e1 (connected to location l) is maximal with respect to the amount of pheromone at all other edges connected to that location l, except the edge that brought the ant to the location,
then, if an ant is at that location l at time t1,
then the next edge the ant will be at some
time t2 > t1 is e1.

If at time t1 an ant is at location 1 and
for every ant arriving at that location 1 at time t1,
the next edge it will be at some time t2 > t1 is e1,
then the amount of pheromone at edge e1 is maximal with respect to the amount of pheromone at all other edges connected to that location l, except the edge that brought the ant to the location.

A formalisation of this property in TTL is as follows:

∀t1,l,l1,e1,e2,i1
[e1≠e2 &
state(γ, t1) |= connected_to_via(l, l1, e1) &
state(γ, t1) |= pheromones_at(e1, i1) &
[∀l2≠l1, e3≠e2 [state(γ, t1) |= connected_to_via(l, l2, e3) ⇒
∃i2 [0≤i2<i1 & state(γ, t1) |= pheromones_at(e3, i2)]]
⇒ ∀a [state(γ, t1) |= is_at_location_from(a, l, e2) ⇒
∃t2>t1 state(γ, t2) |= is_at_edge_from_to(a, e1, l, l1) &
[∀t3 t1<t3<t2 ⇒ is_at_location_from(a, l, e2)]]]]

∀t1, l,l1,e1,e2
[e1≠e2 &
state(γ, t1) |= connected_to_via(l, l1, e1) &
∃a state(γ, t1) |= is_at_location_from(a, l, e2) &
∀a [state(γ, t1) |= is_at_location_from(a, l, e2) ⇒
∃t2>t1 state(γ, t2) |= is_at_edge_from_to(a, e1, l, l1) &
[∀t3 t1<t3<t2 ⇒ is_at_location_from(a, l, e2)]]]
⇒ ∃i1 [state(γ, t1) |= pheromones_at(e1, i1) &
[∀l2≠l1, e3≠e2 [state(γ, t1) |= connected_to_via(l, l2, e3)
⇒ ∃i2 [0≤i2≤i1 & state(γ, t1) |= pheromones_at(e3, i2)]]]]

6. A Simulation Model of Shared Extended Mind

In (Bosse et al., 2004) a simulation model of an ant society is specified in which shared extended mind plays an important role. This model is based on local dynamic properties, expressing the basic mechanisms of the process. In this section, a selection of these local properties is presented, and a resulting simulation trace is shown. In the next section it will be explained how the representation relations specified earlier can be verified against such simulation traces. Here a is a variable that stands for ant, l for location, e for edge, and i for pheromone level.

LP5 (Selection of Edge)
This property models (part of) the edge selection mechanism of the ants. It expresses that, when an ant observes that it is at location l, and there are two edges connected to that location, then the ant goes to the edge with the highest amount of pheromones. Formalisation:
observes(a, is_at_location_from(l, e0)) and neighbours(l, 3) and connected_to_via(l, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(l, l2, e2) and observes(a, pheromones_at(e2, i2)) and e0 ≠ e1 and e0 ≠ e2 and e1 ≠ e2 and i1 > i2 •→» to_be_performed(a, go_to_edge_from_to(e1, l1))

LP9 (Dropping of Pheromones)
This property expresses that, if an ant observes that it is at an edge e from a location l to a location l1, then it will drop pheromones at this edge e. Formalisation:
observes(a, is_at_edge_from_to(e, l, l1)) •→» to_be_performed(a, drop_pheromones_at_edge_from(e, l))

LP13 (Increment of Pheromones)
This property models (part of) the increment of the number of pheromones at an edge as a result of ants dropping pheromones. It expresses that, if an ant drops pheromones at edge e, and no other ants drop pheromones at this edge, then the new number of pheromones at e becomes i*decay+incr. Here, i is the old number of pheromones, decay is the decay factor, and incr is the amount of pheromones dropped. Formalisation:
to_be_performed(a1, drop_pheromones_at_edge_from(e, l1)) and
∀l2 not to_be_performed(a2, drop_pheromones_at_edge_from(e, l2)) and
∀l3 not to_be_performed(a3, drop_pheromones_at_edge_from(e, l3)) and
a1 ≠ a2 and a1 ≠ a3 and a2 ≠ a3 and pheromones_at(e, i) •→→ pheromones_at(e, i*decay+incr)

LP14 (Collecting of Food)
This property expresses that, if an ant observes that it is at location F (the food source), then it will pick up some food. Formalisation:
observes(a, is_at_location_from(l, e)) and food_location(l) •→→ to_be_performed(a, pick_up_food)

LP18 (Decay of Pheromones)
This property expresses that, if the old amount of pheromones at an edge is i, and there is no ant dropping any pheromones at this edge, then the new amount of pheromones at e will be i*decay. Formalisation:
pheromones_at(e, i) and ∀a,l not to_be_performed(a, drop_pheromones_at_edge_from(e, l)) •→→ pheromones_at(e, i*decay)

A special software environment has been created to enable the simulation of executable models. Based on an input consisting of dynamic properties in *leads to* format, the software environment generates simulation traces. An example of such a trace can be seen in Figure 2. Time is on the horizontal axis, the state properties are on the vertical axis. A dark box on top of the line indicates that the property is true during that time period, and a lighter box below the line indicates that the property is false. This trace is based on all local properties identified.

Because of space limitations, in the example situation depicted in Figure 2, only three ants are involved. However, similar experiments have been performed with a population of 50 ants. Since the abstract way of modelling used for the simulation is not computationally expensive, also these simulations took no more than 30 seconds.

As can be seen in Figure 2 there are two ants (ant1 and ant2) that start their search for food immediately, whereas ant3 comes into play a bit later, at time point 3. When ant1 and ant2 start their search, none of the locations contain any pheromones yet, so basically they have a free choice where to go. In the current example, ant1 selects a rather long route to the food source (via locations A-B-C-D-E-F), whilst ant2 chooses a shorter route (A-G-H-F). Note that, in the current model, a fixed route preference (via the attractiveness predicate) has been assigned to each ant for the case there are no pheromones yet. After that, at time point 3, ant3 starts its search for food. At that moment, there are trails of pheromones leading to both locations B and G, but these trails contain exactly the same number of pheromones. Thus, ant3 also has a free choice among location B and G, and chooses in this case to go to B. Meanwhile, at time point 18, ant2 has arrived at the food source (location F). Since it is the first to discover this location, the only present trail leading back to the nest, is its own trail. Thus ant2 will return home via its own trail. Next, when ant1 discovers the food source (at time point 31), it will notice that there is a trail leading back that is stronger than its own trail (since ant2 has already walked there twice: back and forth, not too long ago). As a result, it will follow this trail and will keep following ant2 forever. Something similar holds for ant3. The first time that it reaches the food source, ant3 will still follow its own trail, but some time later (from time point 63) it will also follow the other two ants. To conclude, eventually the shortest of both routes is shown to remain, whilst the other route evaporates. Other simulations, in particular for small ant populations, show that it is important that the decay parameter of the

pheromones is not too high. Otherwise, the trail leading to the nest has evaporated before the first ant has returned, and all ants get lost!

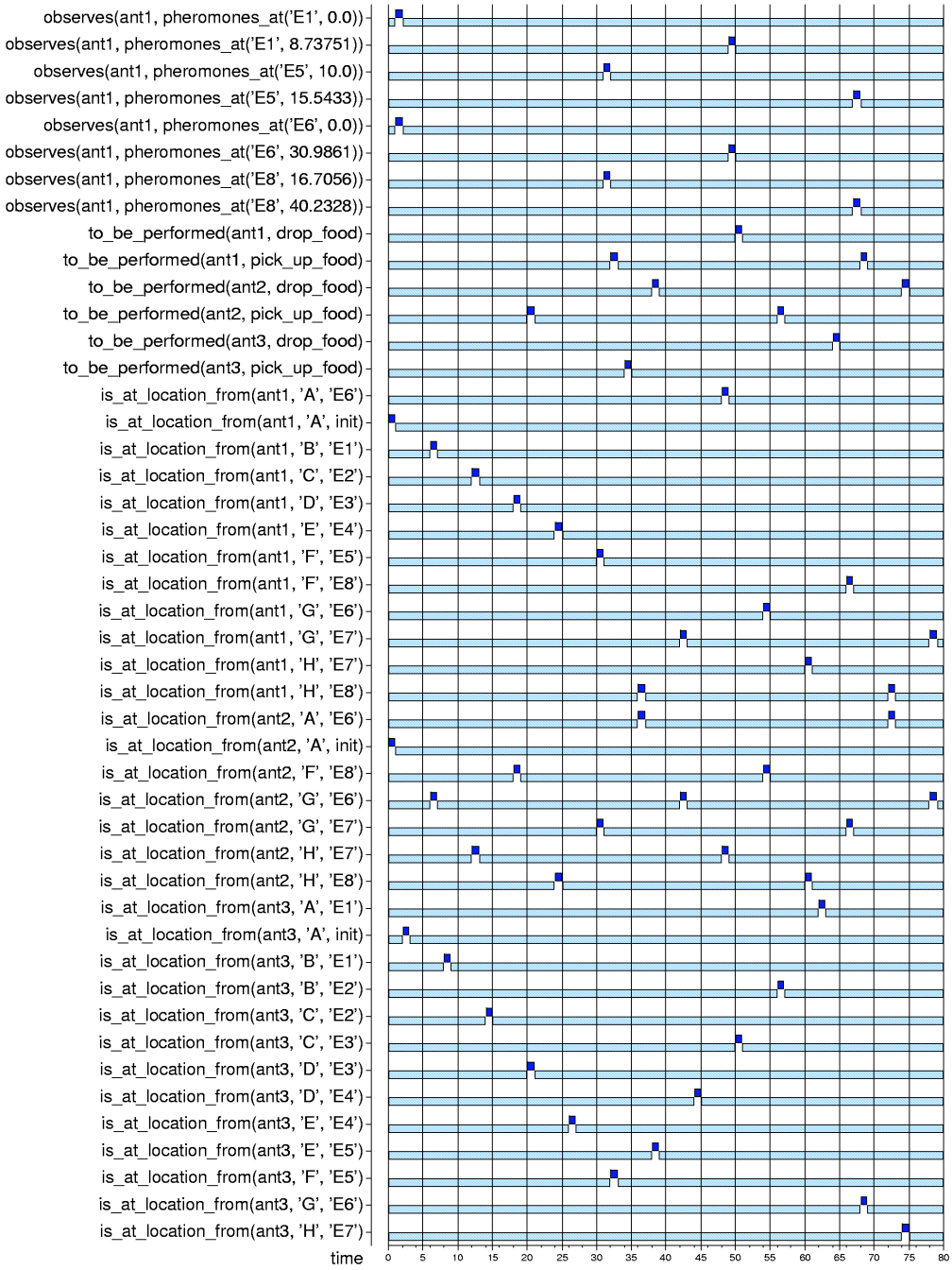


Figure 2. Simulation trace

7. Verification

In addition to the simulation software, a software environment has been developed that enables to check dynamic properties specified in TTL against simulation traces. This software environment takes a dynamic property and one or more (empirical or simulated) traces as input, and checks whether the dynamic property holds for the traces. Traces are represented by sets of Prolog facts of the form

```
holds(state(m1, t(2)), a, true).
```

where m1 is the trace name, t(2) time point 2, and a is a state formula in the ontology of the component's input. It is indicated that state formula a is true in the component's input state at time point 2. The programme for temporal formula checking basically uses Prolog rules for the predicate sat that reduce the satisfaction of the temporal formula finally to the satisfaction of atomic state formulae at certain time points, which can be read from the trace representation. Examples of such reduction rules are:

```
sat(and(F,G)) :- sat(F), sat(G).
sat(not(and(F,G))) :- sat(or(not(F), not(G))).
sat(or(F,G)) :- sat(F).
sat(or(F,G)) :- sat(G).
sat(not(or(F,G))) :- sat(and(not(F), not(G))).
```

Using this environment, the formal representation relations presented in Section 4 and 5 have been automatically checked against traces like the one depicted in Section 6. The duration of these checks varied from 1 to 10 seconds, depending on the complexity of the formula (in particular, the backward representation relation has a quite complex structure, since it involves reference to a large number of events in the history). All these checks turned out to be successful, which validates (for the given traces at least) our choice for the representational content of the shared extended mental state property pheromones_at(e, v). However, note that these checks are only an empirical validation, they are no exhaustive proof as, e.g., model checking is. Currently, the possibilities are explored to combine TTL with existing model checking techniques.

In addition to simulated traces, the checking software allows to check dynamic properties against other types of traces as well. In the future, the representation relations specified in this paper will be checked against traces resulting from other types of ants simulations, and possibly against empirical traces.

8. Discussion

The extended mind perspective introduces a high-level conceptualisation of agent-environment interaction processes. By modelling the ants example from an extended mind perspective, the following challenging issues on cognitive modelling and representational content were encountered:

1. How to define representational content for an external mental state property
2. How to handle decay of a mental state property
3. How can joint creation of a shared mental state property be modelled
4. What is an appropriate notion of collective representational content of a shared external mental state property
5. How can representational content be defined in a case where a behavioural choice depends on a number of mental state properties

These questions were addressed in this paper. For example, modelling joint creation of mental state properties (3.) was made possible by using relative or leveled mental state properties, parameterised by numbers. Each contribution to such a mental state property was modelled by addition to the level indicated by the number. Collective representational content (4.) from a looking backward perspective was defined by taking into account

histories of such contributions. Collective representational content from a forward perspective was defined taking into account multiple parameterised mental state properties, corresponding to the alternatives for behavioural choices, with their relative weights. In this case it is not possible to define representational content for just one of these mental state properties, but it is possible to define it for their combination or conjunction (5.).

The high-level conceptualisation has successfully been formalised and analysed in a logical manner. The formalisation enables simulation and automated checking of dynamic properties of traces or sets of traces, in particular of the representation relations.

For future research, it is planned to make the distinction between extended mind states and other external world states more concrete. In addition, the approach will be applied to several other cases of extended mind. For example, can the work be related to AI planning representations, traffic control, knowledge representation of negotiation, and to the concept of "shared knowledge" in knowledge management?

Acknowledgements

The authors are grateful to Lourens van der Meij for his contribution to the development of the software environment, and to an anonymous referee for some valuable comments on an earlier version of this paper.

References

Bickhard, M.H. Representational Content in Humans and Machines. *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 5, 1993, pp. 285-333.

Bonabeau, J. Dorigo, M. and Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1999.

Bosse, T., Jonker, C.M., Schut, M.C., and Treur, J. Simulation and Analysis of Shared Extended Mind. In: Davidsson, P., Gasser, L., Logan, B., and Takadama, K. (eds.), *Proceedings of the First Joint Workshop on Multi-Agent and Multi-Agent-Based Simulation, MAMABS'04*, 2004, pp. 191-200.

Bosse, T., Jonker, C.M., and Treur, J. Simulation and analysis of controlled multi-representational reasoning processes. *Proc. of the Fifth International Conference on Cognitive Modelling, ICCM'03*. Universitats-Verlag Bamberg, 2003, pp. 27-32.

Clark, A. *Being There: Putting Brain, Body and World Together Again*. MIT Press, 1997.

Clark, A. Reasons, Robots and the Extended Mind. In: *Mind & Language*, vol. 16, 2001, pp. 121-145.

Clark, A., and Chalmers, D. The Extended Mind. In: *Analysis*, vol. 58, 1998, pp. 7-19.

Dennett, D.C. *Kinds of Mind: Towards an Understanding of Consciousness*, New York: Basic Books, 1996.

Griffiths, P. and Stotz, K. How the mind grows: a developmental perspective on the biology of cognition. *Synthese*, vol.122, 2000, pp. 29--51.

Jacob, P. *What Minds Can Do: Intentionality in a Non-Intentional World*. Cambridge University Press, Cambridge, 1997.

Jonker, C.M., and Treur, J. A Temporal-Interactivist Perspective on the Dynamics of Mental States. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 137-155.

Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 191-210.

Kim, J. *Philosophy of Mind*. Westview Press, 1996.

Menary, R. (ed.) *The Extended Mind,* Papers presented at the Conference *The Extended Mind - The Very Idea: Philosophical Perspectives on Situated and Embodied Cognition,* University of Hertfordshire, 2001. John Benjamins, 2004, to appear.

Scheele, M. Team Action: A Matter of Distribution. Distributed Cognitive systems and the Collective Intentionality they Exhibit. *Third International Conference on Collective Intentionality,* 2002.

Appendix A. Simulation Model

LP1 (Initialisation of Pheromones)
This property expresses that at the start of the simulation, at all locations there are 0 pheromones. Formalisation:
start •→» pheromones_at(E1, 0.0) and pheromones_at(E2, 0.0) and pheromones_at(E3, 0.0) and pheromones_at(E4, 0.0) and pheromones_at(E5, 0.0) and pheromones_at(E6, 0.0) and pheromones_at(E7, 0.0) and pheromones_at(E8, 0.0) and pheromones_at(E9, 0.0) and pheromones_at(E10, 0.0)

LP2 (Initialisation of Ants)
This property expresses that at the start of the simulation, all ants are at location A. Formalisation:
start •→» is_at_location_from(ant1, A, init) and is_at_location_from(ant2, A, init) and is_at_location_from(ant3, A, init)

LP3 (Initialisation of World)
These two properties model the ants world. The first property expresses which locations are connected to each other, and via which edges they are connected. The second property expresses for each location how many neighbours it has. Formalisation:
start •→» connected_to_via(A, B, l1) and … and connected_to_via(D, H, l10)
start •→» neighbours(A, 2) and … and neighbours(H, 3)

LP4 (Initialisation of Attractive Directions)
This property expresses for each ant and each location, which edge is most attractive for the ant at if it arrives at that location. This criterion can be used in case an ant arrives at a location where there are two edges with an equal amount of pheromones. Formalisation:
start •→» attractive_direction_at(ant1, A, E1) and … and attractive_direction_at(ant3, E, E5)

LP5 (Selection of Edge)
These properties model the edge selection mechanism of the ants. For example, the first property expresses that, when an ant observes that it is at location A, and both edges connected to location A have the same number of pheromones, then the ant goes to its attractive direction. Formalisation:
observes(a, is_at_location_from(A, e0)) and attractive_direction_at(a, A, e1) and connected_to_via(A, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(A, l2, e2) and observes(a, pheromones_at(e2, i2)) and e1 \= e2 and i1 = i2 •→» to_be_performed(a, go_to_edge_from_to(e1, A, l1))

observes(a, is_at_location_from(A, e0)) and connected_to_via(A, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(A, l2, e2) and observes(a, pheromones_at(e2, i2)) and i1 > i2 •→» to_be_performed(a, go_to_edge_from_to(e1, A, l1))

observes(a, is_at_location_from(F, e0)) and connected_to_via(F, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(F, l2, e2) and observes(a, pheromones_at(e2, i2)) and i1 > i2 •→» to_be_performed(a, go_to_edge_from_to(e1, F, l1))

observes(a, is_at_location_from(l, e0)) and neighbours(l, 2) and connected_to_via(l, l1, e1) and e0 ≠ e1 and l ≠ A and l ≠ F •→» to_be_performed(a, go_to_edge_from_to(e1, l, l1))

observes(a, is_at_location_from(l, e0)) and attractive_direction_at(a, l, e1) and neighbours(l, 3) and connected_to_via(l, l1, e1) and observes(a, pheromones_at(e1, 0.0)) and connected_to_via(l, l2, e2) and observes(a, pheromones_at(e2, 0.0)) and e0 ≠ e1 and e0 ≠ e2 and e1 ≠ e2 •→» to_be_performed(a, go_to_edge_from_to(e1, l, l1))

observes(a, is_at_location_from(l, e0)) and neighbours(l, 3) and connected_to_via(l, l1, e1) and observes(a, pheromones_at(e1, i1)) and connected_to_via(l, l2, e2) and observes(a, pheromones_at(e2, i2)) and e0 ≠ e1 and e0 ≠ e2 and e1 ≠ e2 and i1 > i2 •↠ to_be_performed(a, go_to_edge_from_to(e1, l1))

LP6 (Arrival at Edge)
This property expresses that, if an ant goes to an edge e from a location l to a location l1, then later the ant will be at this edge e. Formalisation:
to_be_performed(a, go_to_edge_from_to(e, l, l1)) •↠ is_at_edge_from_to(a, e, l, l1)

LP7 (Observation of Edge)
This property expresses that, if an ant is at a certain edge e, going from a location l to a location l1, then it will observe this. Formalisation:
is_at_edge_from_to(a, e, l, l1) •↠ observes(a, is_at_edge_from_to(e, l, l1))

LP8 (Movement to Location)
This property expresses that, if an ant observes that it is at an edge e from a location l to a location l1, then it will go to location l1. Formalisation:
observes(a, is_at_edge_from_to(e, l, l1)) •↠ to_be_performed(a, go_to_location_from(l1, e))

LP9 (Dropping of Pheromones)
This property expresses that, if an ant observes that it is at an edge e from a location l to a location l1, then it will drop pheromones at this edge e. Formalisation:
observes(a, is_at_edge_from_to(e, l, l1)) •↠ to_be_performed(a, drop_pheromones_at_edge_from(e, l))

LP10 (Arrival at Location)
This property expresses that, if an ant goes to a location l from an edge e, then later it will be at this location l. Formalisation:
to_be_performed(a, go_to_location_from(l, e)) •↠ is_at_location_from(a, l, e)

LP11 (Observation of Location)
This property expresses that, if an ant is at a certain location l, then it will observe this. Formalisation:
is_at_location_from(a, l, e) •↠ observes(a, is_at_location_from(l, e))

LP12 (Observation of Pheromones)
This property expresses that, if an ant is at a certain location l, then it will observe the number of pheromones present at all edges that are connected to location l. Formalisation:
is_at_location_from(a, l, e0) and connected_to_via(l, l1, e1) and pheromones_at(e1, i) •↠ observes(a, pheromones_at(e1, i))

LP13 (Increment of Pheromones)
These properties model the increment of the number of pheromones at an edge as a result of ants dropping pheromones. For example, the first property expresses that, if an ant drops pheromones at edge e, and no other ants drop pheromones at this edge, then the new number of pheromones at e becomes i*decay+incr. Here, i is the old number of pheromones, decay is the decay factor, and incr is the amount of pheromones dropped. Formalisation:
to_be_performed(a1, drop_pheromones_at_edge_from(e, l1)) and ∀l2 not to_be_performed(a2, drop_pheromones_at_edge_from(e, l2)) and ∀l3 not to_be_performed(a3, drop_pheromones_at_edge_from(e, l3)) and a1 ≠ a2 and a1 ≠ a3 and a2 ≠ a3 and pheromones_at(e, i) •↠ pheromones_at(e, i*decay+incr)

to_be_performed(a1, drop_pheromones_at_edge_from(e, l1)) and to_be_performed(a2, drop_pheromones_at_edge_from(e, l2)) and ∀l3 not to_be_performed(a3, drop_pheromones_at_edge_from(e, l3)) and a1 ≠ a2 and a1 ≠ a3 and a2 ≠ a3 and pheromones_at(e, i) •↠ pheromones_at(e, i*decay+incr+incr)

to_be_performed(a1, drop_pheromones_at_edge_from(e, l1)) and to_be_performed(a2, drop_pheromones_at_edge_from(e, l2)) and to_be_performed(a3, drop_pheromones_at_edge_from(e, l3)) and a1 ≠ a2 and a1 ≠ a3 and a2 ≠ a3 and pheromones_at(e, i) •↠ pheromones_at(e, i*decay+incr+incr+incr)

LP14 (Collecting of Food)
This property expresses that, if an ant observes that it is at location F (the food source), then it will pick up some food. Formalisation:
observes(a, is_at_location_from(l, e)) and food_location(l) •→» to_be_performed(a, pick_up_food)

LP15 (Carrying of Food)
This property expresses that, if an ant picks up food, then as a result it will be carrying food. Formalisation:
to_be_performed(a, pick_up_food) •→» is_carrying_food(a)

LP16 (Dropping of Food)
This property expresses that, if an ant is carrying food, and observes that it is at location A (the nest), then the ant will drop the food. Formalisation:
observes(a, is_at_location_from(l, e)) and nest_location(l) and is_carrying_food(a) •→» to_be_performed(a, drop_food)

LP17 (Persistence of Food)
This property expresses that, as long as an ant that is carrying food does not drop the food, it will keep on carrying it. Formalisation:
is_carrying_food(a) and not to_be_performed(a, drop_food) •→» is_carrying_food(a)

LP18 (Decay of Pheromones)
This property expresses that, if the old amount of pheromones at an edge is i, and there is no ant dropping any pheromones at this edge, then the new amount of pheromones at e will be i*decay. Formalisation:
pheromones_at(e, i) and ∀a,l not to_be_performed(a, drop_pheromones_at_edge_from(e, l)) •→» pheromones_at(e, i*decay)

PART VII DISCUSSION AND FUTURE WORK

CHAPTER 20

Discussion

Discussion

1. Research Overview

As indicated in the introduction of this thesis, the main goal of the research presented was to introduce a novel approach for the analysis of the dynamics of cognitive processes, and to explore its applicability in a variety of cognitive domains. The main benefit of this kind of research is that the nature of the human mind is better understood, which can offer new insights to disciplines like Psychology and Philosophy. A second benefit is that the results of the research can be used to create artifacts that show some human-like intelligence.

To reach the above goal, a general methodology was proposed to analyse cognitive processes in a structured manner. Key elements within this methodology are formalisation of dynamic properties, simulation, experimentation, establishment of interlevel relations, verification and empirical validation. As a supporting vehicle for these different elements, the logical *Temporal Trace Language* (TTL) and the executable LEADSTO language (*Language and Environment for Analysis of Dynamics by SimulaTiOn*) have been introduced. Moreover, two dedicated software environments have been constructed that enable the automated performance of parts of the analysis process.

		4	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
		human reasoning	automated reasoning	multi-rep. reasoning	reasoning about design	human negotiation	human-comp. negotiation	trace conditioning	eating reg. disorders	shared extended mind	collective vs. individual	org. of biological proc.	org. of intracellular proc.	rep. cont. & interplay	rep. cont. & conditioning	rep. cont. & consciousn.	collective rep. cont.
a	informal local properties	X	X	X	X	-	-	X	X	X	X	X	X	X	X	X	X
b	formal executable properties	X	X	X	X	-	-	X	X	X	X	X	X	X	X	X	X
c	simulation traces	-	X	X	X	-	-	X	X	X	X	X	X	X	X	X	X
d	informal non-local properties	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
e	formal non-local properties	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
f	interlevel relations	X	X	X	X	-	-	-	X	X	X	X	X	X	X	X	-
g	verified model	-	X	X	X	-	-	X	X	X	X	X	X	X	X	X	X
h	empirical traces	X	-	-	-	X	X	-	-	-	-	-	-	-	-	-	-
i	validated model	X	-	-	-	X	X	-	-	-	-	-	-	-	-	-	-

Table 1. Overview of which elements of the general methodology are used in the different chapters of this thesis

To explore the applicability of the analysis approach put forward in this thesis, it has been applied in a large number of cognitive domains. In Table 1 it is shown which elements of the methodology have been used in which chapters. Here, the different elements are on the vertical axis. The different chapters of the thesis (except Chapter 1, 2, 3, 20 and 21 that describe, respectively, the Introduction, the LEADSTO language, the TTL language, the Discussion, and Future Work) are on the horizontal axis. A cross indicates that a certain element is used in a certain chapter. Note that the reason for the table not being completely filled is not that some tasks were impossible to perform, but rather that the focus in that particular project was not on those tasks.

As the table shows, all elements of the methodology have been tested extensively. First, in Chapter 4, the methodology has been applied in the analysis of human reasoning processes. Here the focus was on experiments in human reasoning of the pattern 'reasoning by assumption'. In Chapter 5 the same reasoning pattern has been analysed, but here the focus was more on simulation and verification. Chapter 6 (about multi-representational reasoning) follows more or less the same approach as Chapter 5. A difference with Chapter 5 is that the simulation model of Chapter 6 was not generated by means of the LEADSTO software, but by means of the DESIRE modelling framework (cf. Brazier *et al.*, 2002). In Chapter 7, reasoning about the design of complex systems was analysed. Here, again the focus was on simulation and verification. In Chapter 8 and 9, the analysis approach has been applied in the domain of negotiation. In these chapters, the most important aspect was the analysis of empirical data. To this end, a *System for Analysis of Multi-Issue Negotiation* (SAMIN) was introduced. Using SAMIN, several experiments in multi-issue negotiation have been performed and analysed, involving human traces, computer traces, and mixed traces. For the analysis part, SAMIN makes use of automated verification of dynamic properties (by means of the TTL checker tool). In the next two chapters, the focus was on adaptive processes. Here, the proposed methodology for the analysis of cognitive processes has been used in the analysis of two separate adaptive processes: human trace conditioning (Chapter 10) and adaptivity within psychotherapy (Chapter 11). Again, several elements of the methodology have been used, including formalisation, simulation, and establishment of interlevel relations. After that, the analysis approach has been applied in more philosophical domains. In Chapter 12, the principle of shared extended mind for multiple social agents was analysed. In Chapter 13, a comparison and a formal mapping was made between single agent processes and multi-agent processes. In both chapters, all elements of the methodology proved their value, except the last two elements (concerning empirical work), which were not used. In Chapter 14 and 15, the methodology has been used to demonstrate how complex single-agent processes can be modelled as organisations of multiple agents. Here, the idea of interlevel relations was particularly important, but also simulation and verification were performed. Finally, in the last four chapters the analysis approach has been used to define representational content of mental states in a number of cases. Different case studies were used: Chapter 16 addressed the domain of agent-environment interaction, Chapter 17 addressed the neural conditioning of Aplysia, Chapter 18 addressed Damasio's theory of consciousness, and Chapter 19 addressed shared extended mind. In these chapters, again almost all elements of the methodology were used (except the last two). Here, the TTL language turned out to be well suitable for the specification of representation relations in various domains, due to its possibility to specify more complex temporal expressions. Only in Chapter 19 no interlevel relations were established.

2. Evaluation

The previous section clearly indicates that all elements of the general analysis methodology have proved their value in numerous research projects in cognitive domains. Based on the experience in these projects, in this section a number of advantages and possible drawbacks of the approach are formulated. To start with the good news, some positive points of the approach are the following:

First of all, the approach is highly *generic*. Although this thesis was concerned with exploring its applicability in cognitive domains, the approach is not exclusively aimed at analysing cognitive processes. In fact, the approach has been used earlier in several other domains, including biology (Jonker *et al.*, 2002) and social science (Hoogendoorn *et al.*, 2004). The reason for this genericity is that the only domain-specific part of the approach is the ontology that is used for the formalisation in TTL and LEADSTO. And since these languages allow their users to specify new ontologies themselves, the approach is applicable in any domain. This feature distinguishes the approach from more specific approaches that are developed especially for modelling cognition, e.g., ACT-R (Anderson and Lebiere, 1998), SOAR (Laird *et al.*, 1987), and COGENT (Cooper and Fox, 1998). For a brief comparison with these approaches, see the Introduction (Chapter 1).

Next, the approach allows the modeller to analyse process at *different levels of abstraction*. As mentioned above, users can specify ontologies at will (i.e., they are free to choose which concepts they use to describe state properties). This implies that they can analyse processes at any desired level of abstraction. For example, a cognitive process such as reasoning could be described both at a neurological level (in terms of activation of neurons) and at a functional level (in terms of abstract concepts such as assumptions). Especially for cognitive processes this is a useful feature, since these processes are often based on complex internal dynamics. To describe such processes, a too rough model may not give sufficient information. On the other hand, a too detailed model may make it hard to gain insight in the processes. Choosing an appropriate level of abstraction is an important aspect of modelling, and is supported by the proposed methodology. In different chapters of this thesis, indeed various levels of abstraction have been used. For example, in Chapter 14 (about the circulation system) a physiological level was used, in Chapter 15 (intracellular processes) a biochemical level was used, whereas in Chapter 17 (conditioning of Aplysia) and Chapter 18 (Damasio's theory of consciousness) a neurological level was used. In all of the other chapters a functional level of abstraction was used. For an extensive discussion about the topic of abstraction levels, see (Jonker *et al.*, 2002b).

Furthermore, the approach allows the modeller to *combine quantitative with qualitative aspects* of the process under analysis. As mentioned in the Introduction, traditionally two classes of approaches to modelling dynamics are identified: symbolic modelling approaches (e.g., Barringer *et al.*, 1996; Forbus, 1984) and mathematical modelling approaches, usually based on difference or differential equations (e.g., Port and Gelder, 1995). In this light, TTL and LEADSTO can be seen as integrated languages, covering both qualitative concepts and quantitative relations. Several chapters in this thesis have shown that this is an important advantage over traditional approaches. For example, in Chapter 10 it was shown how the temporal dynamics of classical conditioning (which is originally based on differential equations) can be expressed in LEADSTO, thereby allowing the user to combine them with qualitative concepts like instage(reinforcement) within a single model. Also in Chapter 8 and 9

(about negotiation) and in Chapter 11 (about psychotherapy) models are presented that combine qualitative and quantitative aspects. In the other chapters, the focus is on qualitative aspects.

The approach is able to deal with *real-valued time parameters*. As explained in detail in Chapter 2, rules in LEADSTO include four time parameters e, f, g, h, that indicate, respectively, the minimal delay, the maximal delay, the duration of the antecedent, and the duration of the consequent. Unlike in many other modelling approaches (e.g., Barringer *et al.*, 1996; Forbus, 1984), these parameters may be real numbers, which results in more realistic simulations of dynamic processes. This turned out to be beneficial in, for example, Chapter 15. In this chapter, a simulation was performed of the dynamics of intracellular processes. For this case study, several discussions were held with experts in the domain, in order to define specific time parameters for the LEADSTO rules. As a consequence, the resulting simulation traces closely match the corresponding real world processes. Also in both chapters of Part IV (about adaptive processes), an appropriate choice for the time parameters turned out to be essential.

The approach provides support for the *automated analysis of simulated traces*. Since the traces that are generated by the LEADSTO simulation tool can directly be used as input for the TTL checker tool, it is relatively easy for the user to automatically check dynamic properties of simulated traces. Using this technique, simulation models can be verified, i.e., it can be checked whether they satisfy certain expected global properties. This technique has successfully been applied in almost all chapters of this thesis (see Table 1, element g). The duration of such checks varied from one second to a couple of minutes, depending on the complexity of the formula (in particular, the amount of time points). Throughout the whole thesis, the most complex formulae turned out to be the representation relations in Chapter 16 (about agent-environment interaction). Most of these relations resulted in rather complex temporal expressions, sometimes involving up to eight different time points. Using the TTL checker, the verification of such an expression against a single trace took between five and ten minutes.

Using the same technique as above, the approach provides support for the *automated analysis of empirical traces*. Obviously, an important step in the analysis of cognitive processes is to examine empirical data. Within the current approach, this task is facilitated by the formalisation and analysis of empirical traces (i.e., traces that result from human experiments). Since the traces that are needed as input by the TTL checker tool are represented in ASCII format, it is not difficult to convert empirical traces to this format. In order to do this, all that should be done is transcribing the events that occur in the experiments in a formal format (using a domain-specific state ontology) and to assign time values to the events. An example application of this approach is given in Chapter 4, where human reasoning traces are transcribed in a formal notation. Converting empirical traces to this format has two important advantages. First, using the TTL checker, it can be checked whether the traces satisfy certain formal dynamic properties. By checking such properties also for other available traces, all types of traces can be compared automatically. For example, empirical traces can be compared with simulated traces, but also with other empirical traces. Second, using the LEADSTO visualisation tool, the empirical traces can be depicted graphically. This makes it easier to analyse them by hand. Besides in the domain of reasoning (Chapter 4), this validation approach also proved its worth in the domain of negotiation (Chapter 8 and 9).

Using the approach, it is possible to define *interlevel relations* that relate basic mechanisms to more global characteristics of a process. When analysing the dynamics of a (cognitive) process, it is often useful to consider different levels of aggregation. On the one hand, processes can be described in terms of their basic mechanisms. On the other hand, processes can be described in terms of more global characteristics. In general, the global characteristics of the process emerge from its basic characteristics. However, to be able to explain how this emergence works, the exact relationship between both levels of aggregation has to be defined. The establishment of interlevel relations solves this problem by introducing dynamic properties at (possibly many) different levels of abstraction, and logically relating dynamics properties of different levels to each other. This way, it can formally be proven that the dynamic properties at a certain level imply the properties at a higher level. As can be seen in Table 1, the establishment of interlevel relations turned out to be beneficial in several domains, including reasoning (Chapter 4), psychotherapy (Chapter 11) and organisation modelling (Chapter 14 and 15). Especially in this last domain, interlevel relations are important, since they relate the dynamics of different layers in an organisation (e.g., roles, groups, and the overall organisation) to each other. Moreover, interlevel relations are useful for a particular type of explanation of emergence: the *componential explanation*, see (Cummins, 1975; Davies, 2001; Clark, 1997). In this type of explanation, the behaviour of a composite system is explained by referring to the properties of its components and the interaction between them. For example, for the case of the circulatory system (Chapter 14), the fact that mammals exhibit blood circulation may be explained by the fact that they have a well-functioning heart, arteries, and so on.

On a technical level, the language TTL has a higher *expressive power* than standard temporal languages such as LTL and CTL (e.g., Benthem, 1983; Goldblatt, 1992), see also Chapter 3. For example, the possibility of explicit reference to time points and time durations enables modelling of the dynamics of continuous real-time phenomena, such as sensory and neural activity patterns in relation to mental properties (cf. Port and Gelder, 1995). This feature goes beyond the expressive power available in standard linear or branching time temporal logics. The expressiveness of TTL has been exploited most in Part VI about Representational Content. In this part, in order to define appropriate representation relations for mental states, a number of complex temporal expressions had to be formulated, for which TTL turned out to be well suited. Furthermore, the possibility to quantify over traces in TTL allows for specification of more complex adaptive behaviours, such as the property 'exercise improves skill'. This is a relative property in the sense that it involves the comparison of two alternatives for the history. In this thesis, an example of such a property is GP2 in Chapter 10 about conditioning: 'If for trace $\gamma 2$ at time $t2$ peak time $c2$ is more remote than peak time $c1$ for $\gamma 1$ at time $t1$, then at $t2$ in $\gamma 2$ the pending peak level is lower than the pending peak level at $t1$ in $\gamma 1$'. These kinds of relative properties can easily be expressed in TTL, whereas in standard forms of temporal logic different alternative histories cannot be compared.

Despite the above list of advantages, the analysis approach presented in this thesis also has some potential drawbacks. These are considered below:

In a way, the approach is *too generic*. Although the genericity of the approach was mentioned above as a strong point, it can also be viewed as a drawback. Currently, for every new domain, a complete ontology has to be built from scratch. In addition, also the dynamic properties have to be constructed from scratch. This may be a rather

time-consuming and error-prone process. In the future, this drawback could be solved by introducing domain-specific ontologies and certain templates for standard types of properties. See the next section for an elaboration upon these ideas.

As explained in the Introduction, in the current approach the *verification of interlevel relations* is performed *by hand*. Verification of interlevel relations involves the establishment of logical relationships between the dynamic properties at different levels, in such a way that a number of dynamic properties at a certain level together imply a dynamic property at a higher level. At present, this process is performed in a way that is similar to proof methods in mathematics. Obviously, in an ideal situation this process would be automated. However, due to the high complexity of the process, this is currently not feasible. See (Clarke *et al.*, 2000) for an extensive discussion about this topic.

References

Anderson, J.R., and Lebiere, C. (1998). *The atomic components of thought*. Lawrence Erlbaum Associates, Mahwah, NJ.

Barringer, H., Fisher, M., Gabbay, D., Owens, R. & Reynolds, M. (1996). *The Imperative Future:Principles of Executable Temporal Logic*, Research Studies Press Ltd. and John Wiley & Sons.

Benthem, J.F.A.K., van (1983). The Logic of Time: A Model-theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse, Reidel, Dordrecht.

Brazier, F. M. T., Jonker, C.M., and Treur, J. (2002). Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering*, vol. 41, pp. 1-28.

Clark, A. (1997). *Being There: Putting Brain, Body and World Together Aga*in. MIT Press, Cambridge, Mass.

Clarke, E.M., Grumberg, O., and Peled, D.A. (2000). *Model checking*. MIT Press.

Cooper, R., and Fox, J. (1998). COGENT: a visual design environment for cognitive modeling. *Behavior Research Methods, Instruments & Computers*, 30, pp. 553-564.

Cummins, R. (1975). Functional Analysis. *The Journal of Philosophy*, vol. 72, pp. 741-760.

Davies, P.S. (2001). *Norms of Nature: Naturalism and the Nature of Functions*. MIT Press, Cambridge, Mass.

Forbus, K.D. (1984). *Qualitative process theory*. Artificial Intelligence, volume 24, number 1-3, pp. 85-168.

Goldblatt, R. (1992). *Logics of Time and Computation*, 2nd edition, CSLI Lecture Notes 7.

Hoogendoorn, M., Jonker, C.M., Schut, M., and Treur, J. (2004). Modelling the Organisation of Organisational Change. In: Giorgini, P., and Winikoff, M., (eds.), *Proceedings of the Sixth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS'04)*, 2004, pp. 29-46.

Jonker, C.M., Snoep, J.L., Treur, J., Westerhoff, H.V., and Wijngaards, W.C.A. (2002a). Putting Intentions into Cell Biochemistry: An Artificial Intelligence Perspective. *Journal of Theoretical Biology*, vol. 214, pp. 105-134.

Jonker, C.M., Treur, J., and Wijngaards, W.C.A. (2002b). Reductionist and Antireductionist Perspectives on Dynamics. *Philosophical Psychology Journal*, vol. 15, pp. 381-409.

Laird, J.E., Newell, A., and Rosenbloom, P.S. (1987). Soar: an architecture for general intelligence. *Artificial Intelligence*, volume 33, issue 1, pp. 1-64.

Port, R.F., and Gelder, T.J. van (eds.) (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, Mass.

PART VII DISCUSSION AND FUTURE WORK

CHAPTER 21

Future Work

Future Work

The work discussed in this thesis has opened several possibilities for future work. In this section, a number of them are addressed:

A first logical direction for future work would be to apply the analysis approach in *more (cognitive) domains*. Since the approach turned out to be well suitable for the analysis of cognitive processes, it makes sense to broaden the range of applications in this area. In fact, some first steps have already been made at present. Examples of new cognitive phenomena that are currently being investigated using the analysis approach are the concept of free will (Wegner, 2002), and Damasio's somatic marker hypothesis about decision-making (Damasio, 1994, Chapter 8).

As mentioned in the previous section, another future extension would be to introduce *domain-specific ontologies* and *templates for standard types of properties*. As an example of the former, one could think of a specific ontology for the domain of 'reasoning by assumption'. Such an ontology would include standard predicates like assumed(A,S) and rejected(A,S). Likewise, specific ontologies could be created for negotiation, adaptive processes, or multi-agent processes. As an example of the latter, one could think of a template for the set of dynamic properties that describe reactive behaviour. Such a template would roughly have the following format:

reactive_property(I:INFO_ELEMENT, S:SIGN, A:ACTION) ≡
∀γ:Γ ∀t:T
    state(γ,t,input) |== observation_result(I,S) ⇒
        ∃t'>t state(γ,t',output) |== to_be_performed(A)

To instantiate such a property for a certain domain, only the domain-specific sorts INFO_ELEMENT and ACTION have to be defined. Likewise, templates could be defined for dynamic properties that describe, for example, pro-active behaviour, motivation-based behaviour, or representation relations.

Another possible extension to the approach might be the *automated verification of interlevel relations*. This would solve the second drawback that is mentioned in the previous section (i.e., the fact that this is done by hand). Currently, the possibilities are explored to combine TTL with existing model checking techniques. However, due to its high complexity, this is not an easy task. One problem is that the use of real-valued time parameters in principle causes an explosion of the amount of states to be considered. This problem can be dealt with by assuming *finite variability* of state functions (i.e., between any two time points only a finite number of state changes occurs). Relying on this assumption, the complexity of the verification process can be reduced by partitioning time into a finite number of intervals with real-valued durations in which no change occurs. Work in this direction is currently being done by, e.g., (Alur and Dill, 1994; Clarke *et al.*, 2000). Based on this idea, some initial model-checking tools for real-time systems have recently been developed, such as Kronos (Bozga *et al.*, 1998) and UPPAAL (Larsen *et al.*, 1997). However, the main bottleneck in the automated verification of interlevel relations is the establishment of logical relations between dynamic properties at non-local aggregation levels (i.e., between intermediate properties and global properties). Although it is relatively easy to establish interlevel relations between local properties and non-local properties, establishing these relations between intermediate properties and global properties is more complex. In fact, none of the existing techniques is able to perform this process

fully automatically. Therefore, another option is to perform parts of the process automatically, using verification tools such as KIV (Reif, 1995). Future research will have to point out how suitable this option is for the presented methodology.

A final possibility for future research is to design a classification scheme for different types of behaviour. In nature, externally observable agent behaviour can occur in different types, varying from very simple behaviour of an organism to more sophisticated forms. A fundamental challenge within Cognitive Science is to find criteria for the classification of these different types of behaviour. In the future, the analysis approach used in this thesis could contribute to this challenge, for example, by describing different types of behaviour by specific types of TTL formulae. Ideally, this would lead to a set of templates for dynamic properties, in which any type of behaviour as observable in nature could be classified.

References

Alur, R., and Dill, D.L. (1994). A Theory of timed automata. *Theoretical Computer Science*, 126(2), pp. 183-235.

Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., and Yovine, S. (1998). Kronos: A Model-Checking Tool for Real-Time Systems. In: *Proc. of the 10th International Conference on Computer Aided Verification*. Lecture Notes in Computer Science, vol. 1427, Springer Verlag, pp. 546-550.

Clarke, E.M., Grumberg, O., and Peled, D.A. (2000). *Model checking*. MIT Press.

Damasio, A.R. (1994). *Descartes' Error: Emotion, Reason, and the Human Brain*. New York, NY: Grosset/Putnam Press.

Larsen, K.G., Petterson, P., and Yi, W. (1997). UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2), pp. 134-152.

Reif, W. (1995). The KIV approach to software engineering. In: Broy, M., and Jähnichen, S. (eds.), *Methods, languages, and tools for the construction of correct software*. Lecture Notes in Computer Science, vol. 1009, Springer Verlag, Berlin.

Wegner, D.L. (2002). *The illusion of conscious will*. MIT Press.

Samenvatting

1. Cognitief Modelleren

De menselijke geest is ongetwijfeld één van de meest complexe entiteiten die in onze wereld bestaan. Eeuwenlang hebben onderzoekers binnen verschillende disciplines zijn functioneren bestudeerd, variërend van antieke filosofen als Plato tot moderne neurowetenschappers. Er is veel vooruitgang geboekt, bijvoorbeeld dankzij de uitvinding van de Magnetic Resonance Imaging (MRI) technologie, die onderzoekers in staat stelt te kijken welk deel van het brein er actief is gedurende een specifieke hersenactiviteit. Niettemin is in al die jaren slechts een fractie van de mysteries van de menselijke geest ontrafeld. De verzameling van processen die worden uitgevoerd door de menselijke geest wordt soms aangeduid met *cognitie*, en het onderzoeksgebied dat zich bezighoudt met het begrijpen van cognitie staat bekend als *Cognitiewetenschap*. Onderzoekers met verschillende achtergronden werken in dit veld, waaronder linguïsten, antropologen, psychologen, neurowetenschappers, filosofen, en onderzoekers in Kunstmatige Intelligentie (AI). Aangezien de menselijke geest veel verschillende aspecten betrekt, zijn er ook veel manieren om hem te bestuderen. Een recente aanpak, die meer effectief is geworden sinds de snelle ontwikkeling van de Informatica, is *Cognitief Modelleren*. Dit is een methode die als doel heeft om de structuur en de processen van de menselijke geest te bestuderen door ze na te bouwen, zie b.v. (Detje, Dörner, and Schaub, 2003). Overigens zal door dit proefschrift heen een brede interpretatie van het begrip cognitie worden gebruikt, zodat aspecten als emotie en motivatie, maar ook begrippen als bewustzijn worden omvat.

2. Voordelen van Cognitief Modelleren

De voordelen van Cognitief Modelleren kunnen worden geformuleerd vanuit twee verschillende perspectieven. Ten eerste, vanuit het perspectief van de Cognitiewetenschap kan Cognitief Modelleren worden gezien als nuttig om de aard van de menselijk geest en menselijke intelligentie te exploreren. Dit is in overeenstemming met de bovengenoemde claim dat het doel van Cognitief Modelleren is om de structuur en de processen van de menselijke geest te bestuderen door ze na te bouwen. Aldus kunnen, wanneer een zeker aspect van menselijke intelligentie in voldoende detail wordt onderzocht om er een model van te creëren, de experimentele resultaten van zo'n model nieuwe inzichten bieden aan de disciplines die het onderzoek initieerden, zoals Psychologie and Filosofie. Wanneer psychologen bijvoorbeeld een theorie T opstellen die een specifiek redeneerpatroon beschrijft dat mensen in bepaalde omstandigheden gebruiken, dan kan de cognitief modelleur proberen om een (computer) model M van theorie T te maken. Als dit model M vervolgens het door T beschreven gedrag grofweg voorspelt, maar met enkele kleine afwijkingen, dan kunnen de psychologen deze voorspellingen gebruiken om een verfijnde (en hopelijk meer realistische) theorie, T', te creëren. Ten tweede kan het modelleren van de menselijke geest een bron van inspiratie bieden om intelligente artefacten te construeren. Dit wordt vaak benadrukt vanuit een AI perspectief. Een belangrijke uitdaging binnen de AI is immers het bouwen van artefacten die intelligent gedrag vertonen. Dit impliceert dat het veld van Cognitief Modelleren input kan bieden voor AI: wanneer men intelligente artefacten bouwt, kunnen we technieken worden gebruikt die vergelijkbaar zijn met de mechanismen die mensen gebruiken. Zulke technieken kunnen

nuttig zijn, omdat artefacten die een zekere vorm van mensachtige intelligentie bezitten een verscheidenheid aan voordelen hebben boven artefacten die dat niet bezitten. Ze kunnen bijvoorbeeld efficiënter en meer flexibel zijn, een natuurlijker interactie met mensen hebben, en hun gedrag kan worden uitgelegd in begrijpelijker termen. Voor een concreet voorbeeld van een artefact dat al deze voordelen heeft, zou men kunnen denken aan een geautomatiseerde tegenstander in training op basis van simulatie.

Door dit proefschrift heen staat het perspectief van de Cognitiewetenschappen centraal. Aldus is de focus op de formele analyse en modellering van cognitieve processen (zoals redeneren, klassiek conditioneren, en het ontstaan van bewustzijn) met als hoofddoel om de aard van deze processen te bestuderen. Niettemin zal, terwijl dit gedaan wordt, ook het perspectief van de AI soms in ogenschouw worden genomen. Dit betekent dat, bij het modelleren van cognitieve processen, van tijd tot tijd de vraag wordt gesteld hoe de resultaten kunnen worden gebruikt om intelligente artefacten te creëren. Bijvoorbeeld, in Deel III over onderhandelen wordt de dynamiek van menselijke onderhandelings-processen geanalyseerd, en op basis van de resultaten worden er verscheidene suggesties gemaakt om de prestatie van computeronderhandelaars te verbeteren. In deze gevallen is de bijdrage vanuit het AI perspectief echter beperkt tot het aangeven hoe de resultaten van een analyse kunnen worden gebruikt om intelligente artefacten te creëren (b.v., door (formele) requirements voor het gedrag van zulke artefacten te specificeren). De werkelijke implementatie van zulke artefacten daarentegen valt buiten de scope van dit proefschrift.

3. Onderzoeksdoel

In dit proefschrift wordt een nieuwe aanpak voor Cognitief Modelleren geïntroduceerd, die focust op analyse en simulatie van de *dynamiek* van cognitieve processen. Dynamische aspecten spelen een belangrijke rol in de meeste cognitieve processen. Bijvoorbeeld, binnen menselijke redeneerprocessen zijn zulke dynamische aspecten het stellen van redeneerdoelen, het maken van aannames en het evalueren van aannames. Evenzo, binnen het domein van klassiek conditioneren zijn voorbeelden het opbouwen van preparatiesterkte en het reageren op stimuli. Als gevolg hiervan kunnen zulke cognitieve processen niet worden begrepen, gerechtvaardigd of verklaard zonder rekening te houden met deze dynamische aspecten. Daarom is het belangrijkste *onderzoeksdoel* van dit proefschrift om een nieuwe aanpak te introduceren voor de analyse van cognitieve processen, en om zijn toepasbaarheid te onderzoeken in een verscheidenheid van cognitieve domeinen.

4. Onderliggende Principes

De in dit proefschrift geïntroduceerde aanpak is gebaseerd op een aantal onderliggende principes, die hieronder worden samengevat:

- De dynamiek van een proces onthult zich door verschillende *toestanden* die evolueren over de tijd. Een toestand op een gegeven tijdstip is een conjunctie van alle aspecten (of toestandseigenschappen) van de wereld die op dat moment gelden. Een *trace* is een met tijd geïndexeerde sequentie van toestanden. Een trace kan worden gezien als een traject in de muldimensionale ruimte van mogelijke toestanden, d.w.z., het is een specifieke instantie van het te analyseren proces, zie ook (Port and Gelder, 1995).

- Processen kunnen worden gemodelleerd op verschillende niveaus van aggregatie: op een *locaal* niveau, en op *niet-locale* niveaus.
- Op een locaal niveau kunnen processen worden beschreven in termen van hun basale mechanismen. Deze mechanismen, die in dit proefschrift worden beschreven met wat we *locale dynamische eigenschappen* noemen, betreffen de kleinste beschouwde stapjes binnen de conceptualisatie van het te analyseren proces.
- In de praktijk beschrijven locale dynamische eigenschappen het proces meestal vanuit een *intern* perspectief, d.w.z., rekening houdend met mentale toestanden van de agenten (en hun wederzijdse temporele of causale relaties). Het type van de beschouwde mentale toestanden kan variëren. Wanneer dynamische eigenschappen worden uitgedrukt vanuit een *realistisch* perspectief (Kim, 1996) representeren de mentale toestanden bepaalde 'echte' fysieke (b.v. neurologische) toestanden. Wanneer dynamische eigenschappen worden uitgedrukt vanuit een *functionalistisch* perspectief (Kim, 1996) bestaan de mentale toestanden niet noodzakelijk in werkelijkheid, maar kunnen ze veeleer instrumenten zijn om het proces adequaat te beschrijven (b.v., beliefs, desires en intentions, zie Dennett, 1987).
- Vaak kunnen locale dynamische eigenschappen worden uitgedrukt in een *executeerbaar* formaat, d.w.z., voor elke toestand schrijven ze een unieke toekomstige toestand voor. Deze eigenschappen hebben twee voordelen: ze kunnen direct worden gebruikt voor *simulatie* van het te analyseren proces en ze kunnen *grafisch* worden weergegeven (zoals causal graphs of influence diagrams).
- Op een niet-locaal niveau kunnen processen worden beschreven in termen van hun algemene karakteristieken (genaamd *globale dynamische eigenschappen* in dit proefschrift), of in termen van de karakteristieken van delen van het proces (genaamd *tussenliggende dynamische eigenschappen* in dit proefschrift). Beiden typen eigenschappen bestaan doorgaans uit complexe relaties tussen toestanden op verschillende tijdstippen.
- In de praktijk beschrijven globale dynamische eigenschappen het proces meestal vanuit een *extern* perspectief, d.w.z., verwijzend naar het extern observeerbare gedrag van een agent in plaats van naar zijn mentale toestanden.
- Om de graduele formalisering van een proces te vergemakkelijken kunnen alle dynamische eigenschappen worden uitgedrukt in verschillende formaten: in een *informeel* formaat (met behulp van natuurlijke taal), een *semi-formeel* formaat (met behulp van meer gestructureerde natuurlijke taal), en een *formeel* formaat (met behulp van een logische temporele taal). Een informeel formaat is geschikt voor communicatie met domeinexperts, een formeel formaat is geschikt voor verwerking door een computer, en een semi-formeel formaat kan worden gebruikt om het gat tussen de twee te overbruggen.
- Ten gevolge van het uitdrukken van dynamische eigenschappen op verschillende niveaus van aggregatie kunnen bepaalde *logische inter-niveau relaties* worden geïdentificeerd tussen de dynamische eigenschappen van verschillende niveaus. Deze relaties kunnen bijvoorbeeld aangeven dat een aantal locale eigenschappen samen een tussenliggende eigenschapen impliceren, of dat een aantal tussenliggende eigenschappen samen een globale eigenschap impliceren.
- Een belangrijke uitdaging in Cognitief Modelleren is om te verifiëren of de locale eigenschappen die worden gebruikt om de basale mechanismen van een proces te

beschrijven zekere globale eigenschappen doen ontstaan die geacht worden te gelden voor het proces.

- Een andere belangrijke uitdaging in Cognitief Modelleren is om alle dynamische eigenschappen die worden gebruikt om het proces te beschrijven te *valideren* door ze te vergelijken met empirische data.

5. Overzicht van dit Proefschrift

Het formaat van dit proefschrift is een verzameling van artikelen. De meeste hoofdstukken zijn ofwel herdrukken van beoordeelde papers (die elders zijn gepubliceerd), of uitgebreide versies van gepubliceerde papers. De papers zijn onveranderd, met uitzondering van enkele layout-specifieke kwesties. Dit heeft twee belangrijke implicaties. In de eerste plaats is er overlap tussen een aantal hoofdstukken. Ieder hoofdstuk bevat bijvoorbeeld een specifieke sectie waarin de modelleeraanpak opnieuw wordt geïntroduceerd, met speciale aandacht voor de aspecten van de aanpak die relevant zijn voor het domein in kwestie. Ten tweede impliceert het feit dat de meeste hoofdstukken corresponderen met bestaande papers dat elk van hen afzonderlijk kan worden gelezen. Met andere woorden, dit proefschrift heeft geen specifieke leesvolgorde. Echter, aan de lezers die prefereren het gehele proefschrift te lezen wordt aangeraden om de normale volgorde te volgen, beginnend met Hoofdstuk 1 en eindigend met Hoofdstuk 21.

In dit proefschrift zullen verscheidene cognitieve processen in verschillende (deel)domeinen worden bestudeerd. Het proefschrift is gestructureerd aan de hand van zes Delen, elk focussend op een verschillend aspect van cognitieve processen:

I. *Introductie en Basale Technieken*
In Deel I worden het onderwerp van het proefschrift en de algemene onderzoeksmethode geïntroduceerd. Hoofdstuk 1 introduceert het onderwerp van het proefschrift: een methode voor analyse van de dynamiek van cognitieve processen. Hoofdstuk 2 en 3 beschrijven de belangrijkste analysetechnieken die worden gebruikt door het proefschrift heen. Hoofdstuk 2 focust op de LEADSTO taal en software omgeving (voor het specificeren en formaliseren van executeerbare dynamische eigenschappen en simulatie op basis van deze eigenschappen). Hoofdstuk 3 focust op de TTL taal en software omgeving (voor het specificeren en formaliseren van complexe dynamische eigenschappen en checken van deze eigenschappen tegen traces).

II. *Redeneren*
Redeneren is een hogere cognitieve functie, en wordt daarom algemeen beschouwd als een belangrijk onderwerp in de Cognitiewetenschap. In Deel II worden meerdere redeneerpatronen die gangbaar zijn in menselijk redeneren geanalyseerd met behulp van verschillende technieken. Hoofdstuk 4 focust op een specifiek redeneerpatroon, genaamd ‘redeneren met assumpties’. Er wordt een case study uitgevoerd waarin deelnemers een puzzel oplossen door gebruik te maken van dit specifieke redeneerpatroon. Voor de resulterende empirische traces wordt er aangetoond hoe ze kunnen worden geformaliseerd en automatisch kunnen worden geanalyseerd tegen verscheidene dynamische eigenschappen. In Hoofdstuk 5 wordt er een verschuiving gemaakt naar software agenten. Er wordt gedemonstreerd hoe het in Hoofdstuk 4 geïdentificeerde type dynamische eigenschappen kan worden gebruikt voor requirements analyse van een software agent die redeneert met assumpties.

Vervolgens behandelt Hoofdstuk 6 een ander redeneerpatroon: ‘redeneren met meerdere representaties’ en de besturing daarvan. In dit type redeneren kunnen de toestanden van het redeneerproces bestaan uit verschillende representaties, zoals aritmetische, geometrische en materiele representaties. In dit hoofdstuk wordt een simulatiemodel gepresenteerd en een formele analysemethode voor de dynamiek van zulke redeneerpatronen beschreven. Tenslotte richt Hoofdstuk 7 zich op redeneren in een andere context: redeneren binnen het domein van ontwerp van complexe systemen (b.v. software systemen). De dynamiek van dit type redeneren wordt gesimuleerd en geanalyseerd. Er wordt beschreven hoe belangrijke taken in dit type redeneren onder anderen de identificatie van requirements en de toekenning van componenten aan het systeem zijn.

III. *Onderhandelen*

Evenals redeneren is onderhandelen een complexe taak die enige intelligentie vereist. Een verschil met redeneren is dat onderhandelen wordt uitgevoerd door meerdere agenten tezamen, terwijl redeneren is beperkt tot een enkele agent. Dus, voor onderhandelen is ook de dynamiek van de interactie tussen agenten belangrijk, niet alleen de dynamiek van interne mentale processen. In Deel III wordt de dynamiek van (menselijke en geautomatiseerde) onderhandelingsprocessen geanalyseerd. De analyse behelst zowel een locaal perspectief (focussend op de individuele stappen in de onderhandeling) als een globaal perspectief (focussend op het welzijn van het geheel). Hoofdstuk 9 presenteert een generieke software omgeving voor de analyse van gesloten multi-attribuut onderhandeling. Er wordt aangetoond hoe deze omgeving kan worden gebruikt om dynamische eigenschappen te checken tegen onderhandelingstraces. De traces kunnen worden gegenereerd door menselijke onderhandelaars, door software onderhandelaars, of door beiden. Er worden enkele voorlopige resultaten verschaft van experimenten in menselijke multi-attribuut onderhandeling. Om dit werk voort te zetten doet Hoofdstuk 9 verslag van twee experimenten die bijdragen aan de vergelijking van menselijk versus computergedrag in multi-attribuut onderhandeling. Verscheidene voor- en nadelen van menselijke strategieën in vergelijking met computerstrategieën worden getoond. Op basis van de resultaten van de experimenten worden enkele suggesties gemaakt om de prestatie van computer onderhandelaars te verbeteren.

IV. *Adaptiviteit*

Een andere karakteristiek van intelligente, cognitieve agenten is hun vermogen om zich aan te passen aan hun ervaringen in een veranderende omgeving. Zowel mensen als intelligente software agenten kunnen bijvoorbeeld in staat zijn om te leren van hun fouten. Deel IV analyseert de dynamiek van twee voorbeeldprocessen waar adaptiviteit een belangrijke rol speelt: menselijke spoorconditionering en adaptiviteit binnen psychotherapie. In Hoofdstuk 10 wordt adaptiviteit binnen menselijke spoorconditionering onderzocht: hoe leren mensen zich voor te bereiden voor bepaalde taken? Voor dit domein wordt een executeerbaar declaratief logisch model gecreëerd op basis van Machado’s wiskundige model, en zijn een aantal relevante globale eigenschappen geverifieerd tegen de gesimuleerde traces. Hoofdstuk 11 richt zich op een ander type adaptiviteit: adaptiviteit van het menselijk lichaam in het geval van eetstoornissen. De dynamiek van dit proces wordt gesimuleerd, zowel voor goed functionerende situaties als voor verschillende typen van slecht functionerende situaties die corresponderen met de eerste fase van bekende stoornissen zoals anorexia (nervosa), obesitas, and boulimia. Daarnaast

worden deze processen geanalyseerd om termen van globale dynamische eigenschappen en inter-niveau relaties.

V. *Enkele vs. Meerdere Agenten*
Deel V laat zien hoe complexe (cognitieve) processen kunnen worden geanalyseerd vanuit twee verschillende perspectieven: vanuit het perspectief van een enkele agent en vanuit een multi-agent perspectief. Aan de ene kant wordt er nagegaan in hoeverre multi-agent processen die blijk geven van een vorm van collectieve intelligentie kunnen worden geïnterpreteerd als een enkele agent. Aan de andere kant wordt er onderzocht hoe complexe processen binnen een enkele agent kunnen worden gemodelleerd als een organisatie van meerdere agenten. Voordelen van beide perspectieven worden besproken. Eerst introduceert Hoofdstuk 12 het principe van shared extended mind voor meerdere sociale agenten: in de omgeving gecreëerde patronen die de agenten gebruiken als externe mentale toestanden. Dit principe wordt geïllustreerd door een case study in mierengedrag, waarvan de dynamiek wordt geformaliseerd en gesimuleerd in termen van dynamische eigenschappen. Daarna pakt Hoofdstuk 13 de vraag aan in hoeverre zo'n complex proces van meerdere agenten kan worden geïnterpreteerd als een proces van een enkele agent. Voor het voorbeeldproces van Hoofdstuk 12 wordt laten zien hoe het kan worden geconceptualiseerd en geformaliseerd op twee verschillende manieren: vanuit het perspectief van een enkele agent en vanuit een multi-agent perspectief. Verder wordt er aangetoond hoe een ontologische mapping tussen de twee formaliseringen formeel kan worden gedefinieerd, en hoe deze mapping kan worden uitgebreid naar een mapping van dynamische eigenschappen. Vanuit een ander perspectief laten Hoofdstuk 14 en 15 zien hoe een complex proces van een enkele agent (in dit geval binnen de biologie) kan worden gemodelleerd als een organisatie van meerdere agenten. De aanpak wordt in Hoofdstuk 14 geïllustreerd voor het geval van de bloedsomloop van zoogdieren, en in Hoofdstuk 15 voor het geval van het eencellige organisme *E.coli*. In beide hoofdstukken worden verschillende componenten van het te analyseren systeem gemodelleerd als rollen in een organisatie. Voorts worden, in overeenstemming met de generieke modelleeraanpak die door dit proefschrift heen gebruikt wordt, dynamische eigenschappen geïdentificeerd voor verschillende niveaus van het organisatiemodel, en worden inter-niveau relaties expliciet gemaakt.

VI. *Representationele Inhoud*
In Deel VI wordt ingegaan op het filosofische concept 'representationele inhoud' voor mentale toestanden. Het toekennen van representationele inhoud aan mentale toestanden is een fundamentele uitdaging binnen de AI, Filosofie en Cognitiewetenschap. Eigenlijk is de vraag hier: 'wat betekent het dat een (kunstmatige of echte) agent een mentale toestand heeft?'. Voor een aantal mentale toestanden in verschillende domeinen wordt aangetoond hoe hun representationele inhoud op een precieze manier kan worden gedefinieerd en geformaliseerd. In Hoofdstuk 16 wordt eerst het concept representationele inhoud kort geïntroduceerd. Daarna wordt de toepasbaarheid van een aantal bestaande aanpakken voor representationele inhoud onderzocht voor een specifieke case study. Deze case study betreft een voorbeeld waar een extensieve interactie tussen agent en omgeving plaatsvindt, hetgeen traditioneel wordt gezien als een geval waar het moeilijk tot onmogelijk is om representationele inhoud te definiëren. Daarna past Hoofdstuk 17 het idee van representationele inhoud toe op de neurale mechanismen van klassiek conditioneren. In dit hoofdstuk wordt een simulatiemodel beschreven van het

neurale conditioneringsmechanisme van de zeeslak Aplysia, wat een van de meest simpele (en daarom best begrepen) conditioneringsmechanismen in bestaande organismen is. Voor een aantal interne toestanden van dit model wordt er aangetoond hoe de representationele inhoud formeel kan worden gedefinieerd. Nog een ander domein betreffend, past Hoofdstuk 18 het idee van representationele inhoud toe op de mentale processen die leiden tot de geboorte van bewustzijn. In dit hoofdstuk wordt een formeel model verschaft van Damasio's theorie over kernbewustzijn, en wordt voor een aantal belangrijke concepten in zijn theorie de representationele inhoud formeel gedefinieerd. Tenslotte laat Hoofdstuk 19 zien hoe het idee van representationele inhoud kan worden gecombineerd met het principe van shared extended mind (zie Hoofdstuk 12), daarbij een notie van collectieve representationele inhoud introducerend. Voorstellen voor collectieve representationele inhoud worden geformaliseerd en automatisch geverifieerd.

VII. *Discussie en Toekomstig Werk*

Om het proefschrift af te sluiten worden in Deel VII zijn belangrijkste bijdragen samengevat en worden enkele toekomstige onderzoeksmogelijkheden besproken. In Hoofdstuk 20 wordt geëvalueerd in welke mate het onderzoeksdoel - het introduceren van een nieuwe aanpak voor de analyse van de dynamiek van cognitieve processen en het onderzoeken van zijn toepasbaarheid - is bereikt. Er wordt aangetoond dat alle elementen van de methode uitvoerig zijn getest, en dat zij een aantal belangrijke voordelen biedt ten opzichte van bestaande aanpakken. Tevens worden enkele potentiële bezwaren van de aanpak besproken. Tenslotte worden in Hoofdstuk 21 enkele mogelijke richtingen voor toekomstig werk aangegeven.

Referenties

Dennett, D.C. (1987). *The Intentional Stance*. MIT Press, Cambridge, Massachusetts.

Detje, F., Dörner, D., and Schaub, H. (eds.) (2003). *The Logic of Cognitive Systems: Proceedings of the Fifth International Conference on Cognitive Modeling, ICCM'03*. Universitäts-Verlag Bamberg.

Kim, J. (1996). *Philosophy of Mind*. Westview Press.

Port, R.F., and Gelder, T.J. van (eds.) (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, Mass.

SIKS Dissertation Series

1998

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

2000

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie;
een procesbenadering en actorperspectief.

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval.

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peek (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coup, (EUR)
Sensitivity Analyis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management

2001

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schonhage (VU)
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 Pieter Jan 't Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models,
Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)
Modeling and Simulating Work Practice
BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VUA)
Knowledge Management:
The Role of Mental Models in Business Systems Design

2002

2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel(KUB)
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (Uva)
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling

2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy

2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)
Learning Search Decisions

2004

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)
Learning from Design: facilitating multidisciplinary design teams

2005

2005-01 Floor Verdenius (UVA)
Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM))
AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)
Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UVA)
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)
Adaptive Game AI

2005-07 Flavius Frasincar (TUE)
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)
A Model-driven Approach for Building Distributed Ontology-based
Web Applications

2005-09 Jeen Broekstra (VU)
Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UVA)
Explaining Behaviour: Using Qualitative Simulation in Interactive
Learning Environments

2005-11 Elth Ogston (VU)
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)
Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)
Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)
Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)
Software Specification Based on Re-usable Business Components

2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks